VAX-11/780
Microprogramming Tools
User's Guide
Order No. AA-H306B-TE

March 1982

First Printing, June 1979 Second Printing, March 1982

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

The software described in this document is furnished under a license and may by used or copied only in accordance with the terms of such license.

No responsibility is assumed for the use or reliability of software on equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.

Copyright © 1982 by Digital Equipment Corporation.
All Rights Reserved.

Printed in U.S.A.

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DECsystem-10	PDT
DECUS	DECSYSTEM-20	RSTS
DIGITAL	DECwriter	RSX
PDP	DIBOL	VMS
UNIBUS	EduSystem	VT
VAX	IAS	DIGITAL Logo
DECnet	MASSBUS	ZKDSR

TABLE OF CONTENTS

CHAPTER	1	INTRODUCTION	
CHAPTER	2	ASSEMBLING YOUR MICROPROGRAM	
	2.1	PROGRAM STRUCTURE	-1
	2.1.1	The Bit Numbering 2	-2
	2.1.2	The Program Radix 2	-2
	2.1.3	Memories	-2
	2.1.4	The Program Title 2	-3
	2.1.5	The Table of Contents 2	-3
	2.1.6	Listing Pagination 2	-3
	2.1.7		-4
	2.2		-4
	2.2.1		-4
	2.2.2		- 5
	2.2.3	Qualifiers	
	2.2.3.1		- 5
	2.2.3.2	The .ADDRESS and .NEXTADDRESS	•
		Qualifiers 2	-5
	2.2.3.3	The .VALIDITY Qualifier 2	-6
	2.2.4		-6
	2.3	EXPRESSIONS	
	2.3.1	Numbers	
	2.3.2	Expression-Names 2	
	2.3.3	Function Calls	
	2.3.4	Value-names 2	
	2.3.5	Field Contents Indicators 2	
	2.3.6	Predefined Symbol Names 2	
	2.4		-9
	2.4.1		-9
	2.5	THE MACRO-BODY	
	2.5.1	Parameters	
	2.6	MICROINSTRUCTIONS	-10
	2.6.1	MICROINSTRUCTIONS	-11
	2.7	THE MICROWORD	-11
	2.8	THE ADDRESS SPACE	-11
	2.9		-11
	2.9.1	Sequential Allocation	-12
	2.9.2		-12
	2.9.2.1		-12
	2.9.2.2	Indicating a bit that can be Ø or 1 . 2	
	2.9.2.3		-13
	2.9.2.4		-13
	2.9.2.5		-13
	2.10		-13
	2.10.1		-13
	2.10.2		-14
	~ • 10 • 4	LIGHT COMMUNICACION	

	2.11	CONDITIONAL ASSEMBLY 2-	14
	2.11.1	The Conditional Assembly Keywords 2-	
	2.11.2	Conditional Assembly Blocks 2-	
	2.12	SETTING AND CHANGING EXPRESSION-NAMES 2-	
	2.13	LIST CONTROLS 2-	
	2.13.1	The List Control Counters 2-	
	2.14	THE VAX 11/780 DEFINITION LANGUAGE 2-	16
	2.15	USER INTERFACE 2-	
CHAPTER	3	LOADING A MICROPROGRAM	
	3.1	FUNCTIONS	1
	3.1.1	Verifying the Installation of the	
		Extended WCS Board3-	1
	3.1.2	Initializing the Extended WCS 3-	
*	3.1.3	Setting the Starting Address	
	3.1.4	Loading the Microprogram3-	
	3.1.5	Logging the WCS Load	
	3.2	VERIFYING THAT THE LOADING WAS SUCCESSFUL . 3-	Δ
	3.2.1	Using the Sample Program3-	
	3.2.2	Sequencing with the Debugger 3-	
	3.3	USER INTERFACE	5
	3.4	PROGRAM BEHAVIOR	7
	3.4.1	VAX 11/780 WCS Architecture Description . 3-	7
	3.4.2	VMS Operating System Support 3-	à
	3.5	ERROR MESSAGES	ĺØ
CHAPTER	4	EXECUTING A MICROPROGRAM	
	4.1	EXTENDED FUNCTION CALL 4-	1
	4.1.1	Exceptions	
	4.1.2	Setting the System Control Block Vector . 4-	2
	4.1.3	Patching the Entry Vector 4-	
APPENDIX	(A	VAX 11/780 FIELD AND MACRO DEFINITIONS	
APPENDIX	К В	SAMPLE MICROPROGRAM FOR SYSTEM REVISION > 7	
	B.1	THE INPUT FILE (.MIC)	2
	B. 2	THE LISTING FILE (.MCR)	
	B.3	THE OBJECT FILE (.ULD)	34
APPENDIX	С	SAMPLE MICROPROGRAM FOR SYSTEM REVISION < 7	
	C.1	THE INPUT FILE (.MIC)	2
	C.2	THE LISTING FILE (.MCR)	5
	C.3	THE LISTING FILE (.MCR)	34
		THE TEST PROGRAM	

PREFACE

Manual Objectives

This manual describes the use of the tools available for the VAX 11/780 WCS user. The appendices contain the VAX 11/780 Definition Language, and a sample program expressed in this language.

Intended Audience

This manual is intended for assembly language programmers and hardware engineers. The reader is assumed to be familiar with microprogramming and the characteristics of the VAX CPU architecture.

Structure of this Document

This manual describes the process of assembling, loading, and executing a microprogram.

Associated Documents

Information on the MICRO2 assembler is given in the following document:

MICRO2 User's Guide/Reference Manual (AA-H531A-TE)

Information on the VAX 11/780 Data Path is given in the following document:

VAX 11/780 Data Path Description (AA-H307A-TE)

Information on the VAX 11/780 software is given in the following document:

VAX 11/780 Software Handbook

The VAX/VMS command language and environment are described in the following document:

VAX/VMS Command Language User's Guide (AA-D023A-TE)

The text editing capability is described in:

VAX-11 Text Editing Reference Manual (AA-D029A-TE)

Information on the VAX 11/780 Hardware is given in the following documents:

VAX 11/780 Hardware Handbook VAX 11/780 Architecture Handbook

Conventions Used in this Document The conventions used in this document are the same as those used in the VAX/VMS Command Language User's Guide (AA-D023A-TE).

CHAPTER 1

INTRODUCTION

The VAX-11/780 Extended Writable Control Store consists of 2048 words occupying addresses 1800 through 1FFF, 1024 words occupying addresses 1000 through 1FFF when G&H is present. Each word contains 96 bits plus three parity bits. User microprograms that enhance a machine for specific applications execute in the Extended WCS. The process of writing, loading, and executing microprograms is diagrammed below.

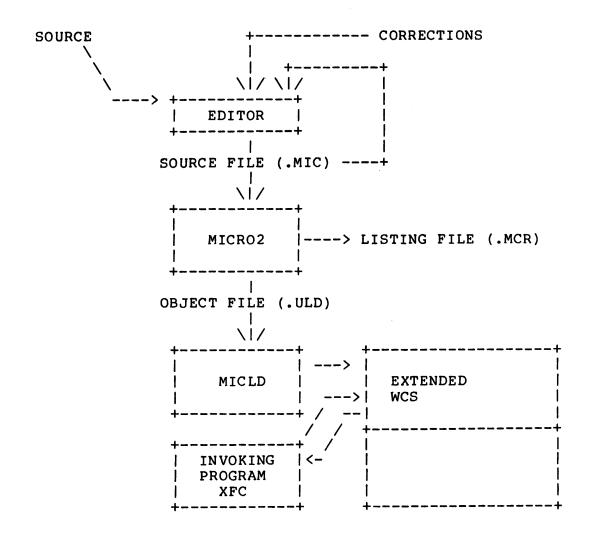


Figure 1-1

INTRODUCTION Page 1-2

The first step in microprogramming the WCS is the creation of the microprogram. To do this, you write the microprogram in MICRO2 source language and create a source file (.MIC). Then, you assemble the microprogram to create a listing file (.MCR) and an object file (.ULD). MICRO2 detects as many errors as possible in the source microprogram. You correct the errors and reassemble until you are satisfied with the resulting microprogram.

When you are ready to test the microprogram, you load it into the WCS using the microprogram loader, MICLD. You can then transfer to the microprogram by executing an XFC instruction in a main memory program.

This manual describes assembling, loading, and executing a microprogram for the VAX 11/780 WCS. Some information on debugging a microprogram can be found in Appendixes C and D of the VAX 11/780 Data Path Description (AA-H307A-TE).

A macro language for microprogramming the VAX 11/780 is given in Appendix A. This language, called the VAX 11/780 Predefinition language, defines the fields of the microword and a set of macros for performing the logical functions associated with microprogramming the VAX 11/780. A VAX 11/780 microprogram written in this definition language is given in Appendix B.

CHAPTER 2

ASSEMBLING YOUR MICROPROGRAM

MICRO2 is a general purpose tool, written in BLISS, that executes under user control in the VAX/VMS environment. The MICRO2 language lets you express the actions of a microprogram symbolically.

The MICRO2 assembler translates a microprogram written in its source language to the bit representation that is loaded into the Extended WCS. In doing this, it performs syntactic checks on the program and issues messages if the source microprogram is not valid.

Further, MICRO2 allocates any microwords that you do not specifically allocate. You can allocate a microword absolutely, specify a constraint on its allocation (such as the two lowest order bits of the address must be zero), or leave the allocation to MICRO2.

The following sections provide a quick reference to the MICRO2 language and its use. A complete description of the MICRO2 assembler is given in a separate document:

MICRO2 User's Guide/Reference Manual (AA-H531A-TE)

The MICRO2 User's Guide/Reference Manual is a tutorial, which contains many examples of the use of the MICRO2 language.

2.1 PROGRAM STRUCTURE

The MICRO2 assembler is a line-oriented processor, which accepts a sequence of input lines written in MICRO2 source language and produces a listing file and an object module.

The input to MICRO2 is a source program. A MICRO2 source program can contain one or more memories. The bit-numbering direction and the program radix apply to the entire program.

2.1.1 The Bit Numbering

The .LTOR and .RTOL keywords define the way in which the bits of a microword are numbered, so that MICRO2 knows whether to count from the left end or the right end of the word to locate a bit position. The form of the bit-numbering keyword-line is the keyword itself, namely:

.LTOR

The .LTOR keyword directs MICRO2 to consider the bits of the microword numbered from left to right. The .RTOL keyword directs MICRO2 to consider the bits numbered from right to left.

If no bit-numbering keyword is given, then .LTOR is assumed.

MICRO2 uses the first .LTOR or .RTOL keyword it finds to establish the direction in which the bits are numbered and ignores any subsequent bit-numbering keywords in the program.

2.1.2 The Program Radix

The program radix is set by either the .OCTAL or .HEXADECIMAL keywords. The form of the .OCTAL and .HEXADECIMAL keyword-line is simply the keyword itself, as follows:

.OCTAL

.HEXADECIMAL

MICRO2 uses the first program radix keyword it find and ignores any subsequent program radix keywords in the program.

2.1.3 Memories

A program can include data for as many as 26 memories. Except for the bit-numbering convention and program radix, which can be specified only once in a program, all other constructs are considered to belong to a memory. Each memory has its own address-space, word-width, field- and macro-definitions, and microinstructions.

The beginning of a section of the memory is specified by a memory-indicator keyword. The memory-indicator keywords are as follows:

.xCODE

where x is any one of the letters in the alphabet

ex. .UCODE .ACODE

The identification, definitions, and instructions following that memory-indicator and up to the end of the program or another memory-indicator are associated with the specified memory.

2.1.4 The Program Title

The .TITLE keyword-line supplies a title for MICRO2 to use as part of the heading of the output listing. MICRO2 reproduces the quoted string following the .TITLE keyword as part of the first line of the output listing. The .TITLE keyword-line has the following form:

.TITLE "title-string"

2.1.5 The Table of Contents

The .TOC keyword-line supplies a subtitle and adds an entry to the table of contents. MICRO2 reproduces the text given in quotes following the .TOC keyword as part of the second line of the page heading of the output listing. The .TOC keyword-line has the following form:

.TOC "text-string"

2.1.6 Listing Pagination

The .PAGE keyword-line indicates a new listing page and, optionally, provides a table of contents entry and a subtitle.

To simply indicate a new page for the output listing, include the .PAGE keyword without any text string, as follows:

. PAGE

To start a new page, add a subtitle, and make an entry in the table of contents, add a text string to the .PAGE keyword-line, as follows:

.PAGE "text-string"

2.1.7 Comments

Comments can be included anywhere in the program. A comment begins with a ";" character and ends at the end of the line.

2.2 FIELD DEFINITIONS

A field-definition consists of a name followed by the separator '/=' followed by the position of the bits within the word to be associated with the field name followed by a list of one or more qualifiers. The form is:

```
field-name /= < left-bit:right-bit > { , qualifier ... }
```

A single bit field can be expressed by including only the left-bit within the angle brackets. Qualifiers can be omitted.

2.2.1 Names

A field-name can be any valid MICRO2 name. MICRO2 allows a name to be made up of characters from the following set:

A B C	Z	Upper case letters
a b c	z	Lower case letters
		Numbers
!		Exclamation mark
#		Hash mark
&		Ampersand
(Left parenthesis
)		Right parenthesis
<		Left angle bracket
< >		Right angle bracket
*		Asterisk
+		plus sign
_		minus sign
•		period
?		question mark
		Underscore
_		Space and tab
		-

2.2.2 Field Position

The left-bit and right-bit are decimal numbers that identify the beginning and end bits of the field in the microword. If you specified right-to-left bit numbering, then the left-bit must be greater than or equal to the right-bit. If you specified left-to-right bit numbering, then the left-bit must be less than or equal to the right-bit.

2.2.3 Qualifiers

Qualifiers are used to establish a default for a field, to identify the field as one that can contain a label, to designate a field to be used for a parity bit, and to associate the setting of a field with the condition of other fields within the microword.

2.2.3.1 The .DEFAULT Qualifier - The .DEFAULT qualifier specifies a value that MICRO2 can use for a field when the field is not explicity set. The form is:

.DEFAULT = expression

In forming a microword, MICRO2 begins with a word consisting of all zeroes, sets the fields explicitly set in the microinstruction, and then applies defaults. MICRO2 uses a default for a field if and only if no bit of the field is set explicitly.

In applying defaults, MICRO2 uses the order in which the fields are specified in the microprogram.

2.2.3.2 The .ADDRESS and .NEXTADDRESS Qualifiers - MICRO2 requires that the jump field be identified by either an .ADDRESS or .NEXTADDRESS qualifier. A field defined with the .ADDRESS or .NEXTADDRESS qualifier can be set to the value of any label in the program. The form is simply the keyword, namely:

.NEXTADDRESS

. ADDRESS

In addition to designating the associated field as a jump field, the .NEXTADDRESS qualifier specifies that the default for the field is the value of the address associated with the next microinstruction given in the program.

2.2.3.3 The .VALIDITY Qualifier - The .VALIDITY qualifier lets you make assertions about the conditions under which a field can be legally set. The form is:

.VALIDITY = expression

The .VALIDITY qualifier associates a validity expression with a field. If the validity expression is not satisfied when the field is set in a microword, MICRO2 produces a warning message.

2.2.4 Value Definitions

A value-definition associates a name with a particular value of a particular field.

Value-definitions follow a field definition. A value-definition conists of a value-name followed by the separator '=' followed by the value to be equated with that name. The value-definition can also have its own .VALIDITY expression. Thus, the form is:

value-name=value,.VALIDITY=exp

A value-name is any valid MICRO2 name, as defined in Section 2.2.1. The .VALIDITY expression is optional.

2.3 EXPRESSIONS

An expression in MICRO2 is enclosed in angle brackets. An expression can be any of the following:

A number

An expression name

A function call

A field value name

A field contents indicator

A predefined symbol

The following sections consider each of these cases in detail.

2.3.1 Numbers

MICRO2 recognizes integers or decimal numbers. An integer is interpreted according to the program radix. A number with a decimal point is always interpreted as a decimal number. The program radix is set by either the .OCTAL or the .HEXADECIMAL keyword. If a program radix is not given, then an octal radix is assumed.

2.3.2 Expression-Names

An expression-name is defined by the .SET keyword as follows:

.SET/expression-name = <expression>

2.3.3 Function Calls

MICRO2 provides functions for comparison, arithmetic, and Boolean operations. Also, MICRO2 provides functions to detect parity, shift, and select a case from a set of choices.

The functions are given in the following table:

<u>Function</u>	Value
Comparison .EQL[opl,op2,] .NEQ[opl,op2,] .GTR[opl,op2,] .GEQ[opl,op2,] .LSS[opl,op2,] .LEQ[opl,op2,]	<pre>l if opl=op2= l if opl<>op2 and op2<>op3 and l if opl>op2> l if opl>=op2>= l if opl<=op2< l if opl<=op2<</pre>
Arithmetic .MAX[opl,op2,] .MIN[opl,op2,] .SUM[opl,op2,] .PROD[opl,op2,] .DIFF[opl,op2] .QUOT[opl,op2] .MOD[opl,op2]	Value of largest operand Value of smallest operand opl+op2+ opl*op2* opl-op2 opl/op2 (truncated) remainder of opl/op2
Boolean	
.NOT[op] .AND[op1,] .OR[op1,] .XOR[op1,] .NAND[op1,] .NOR[op1,]	Boolean complement of op Boolean 'and' of operands Boolean 'or' of operands Boolean 'xor' or operands Boolean complement of the 'and' Boolean complement of the 'or' Boolean complement of the 'xor'
Miscellaneou	ıs
.PARITY[opl,op2,]	If operands contain an even number of 1's, then 1 else 0
.SHIFT[op1,op2]	If op2 is positive, then shift opl left op2 places else shift opl right op2 places
.CASE[op1]OF[op2,]	The (opl-th + 1) operand of the list. That is, if opl is 0, the first op2 is used. Up to 32 choices can be given.
.SELECT[{opl,op2,}]	The first op2 for which opl is true

The operands of a function can be expressions.

2.3.4 Value-Names

Since a value-name is only defined for a specific field, it must be qualified by the field-name when used in an expression as follows:

field-name/value-name

2.3.5 Field Contents Indicators

The contents of a field can be designated in an expression by giving the field-name followed by a slash. For example, to find out if the current contents of field B contains the value 4, you write the following expression:

.EQL[,<4>]

2.3.6 Predefined Symbol Names

The following symbols are predefined in MICRO2, as follows:

Symbol

Meaning

. (period)

The address of the current microinstruction

2.4 MACROS

The macro capability of MICRO2 permits the definition of a representation for a microprogram at a higher level than the basic field-value pairs. Once the fields of your microword are defined, a set of macros that set groups of fields appropriately for certain operations can be specified. Macros cannot generate more than one microinstruction.

2.4.1 The Macro-Name

Macro-names are formed using the set of characters given in Section 2.2.1. In addition to these characters, MICRO2 recognizes square bracket pairs and commas in macro-names as indicators of the number and position of the macro parameters.

The number and position of parameters in a macro-name are an integral part of the name. (That is, the macro ABC[][] is not the same as ABC[,].)

2.5 THE MACRO-BODY

The macro-body consists of any combination of field-settings and macro-calls separated by commas. When a macro is used in a microinstruction, MICRO2 replaces the macro-name by the macro-body associated with that name.

2.5.1 Parameters

Square brackets and commas indicate parameters in the macro-name. The character "0" followed by a decimal integer in the macro-body indicates the position of the parameter in the macro-body. This character pair is called a parameter-designator.

The decimal integer in the parameter-designator refers to the position, numbering from left to right, of the parameter in the name.

2.6 MICROINSTRUCTIONS

The microinstructions describe the processing to be performed by the microprogram. These microinstructions are expressed in terms of the field- and macro-names defined.

For each microinstruction, MICRO2 translates names into the appropriate sequence of bits and creates the associated microword. The microinstruction contains the information MICRO2 needs to set the bits of the microword.

A microinstruction begins with an absolute address assignment, one or more labels, or both. Following this optional information, a sequence of field-settings and/or macro-calls is given separated by commas.

That is, the form of the microinstruction is:

Both the address and label can be omitted.

2.6.1 Continuing A Microinstruction

If a microinstruction occupies more than one line, the separator character ',' must be as the last non-blank character of all lines except the last line. For purposes of this discussion, the end of the line is assumed to be either the ';' character, which begins a comment, or the actual end of line. Thus the last non-blank character of a line means the last non-blank before the ';' or end of line.

2.7 THE MICROWORD

MICRO2 creates a microword in the following way:

- 1. MICRO2 begins with a word of the specified length in which each bit has a value of zero and a status of unset.
- MICRO2 then fills in all the fields that are explicitly set in the microinstruction.
- 3. Then, MICRO2 sets any fields that have an associated default and that contain only unset bits.
- 4. Then, MICRO2 evaluates any VALIDITY expressions.
- 5. Finally, MICRO2 performs any parity adjustment indicated.

2.8 THE ADDRESS SPACE

The .REGION keyword determines the address-space. The .REGION keyword is followed by one or more pairs of address limits, as follows:

.REGION/low-bound, high-bound...

Low-bound and high-bound are expressions whose values are interpreted according to the program radix. An address-space thus can consist of any number of address-ranges. An address-range is specified by the low-bound and high-bound.

Any number of .REGION keyword-lines can be given. MICRO2 allocates the microinstructions following a .REGION up to the next .REGION keyword (or the end of memory) in the specified address space.

If a .REGION keyword-line is not given at the beginning of a memory, MICRO2 assumes that the address space begins at 0 and ends at MAXPC, the highest available address for the given architecture.

2.9 SPECIFYING THE METHOD OF ALLOCATION

Within the specified address space, either sequential or random allocation (or some combination of both) can be used.

2.9.1 Sequential Allocation

In sequential mode, MICRO2 allocates a microinstruction by taking the address of the previous microinstruction and adding 1.

MICRO2 begins allocating with the first address in the address space defined by the .REGION keyword and continues incrementing until it reaches either an absolute address assignment or the end of an address-range.

When it reaches an absolute address, MICRO2 uses that address for the associated microinstruction and as the new base for incrementation.

When MICRO2 uses the last instruction in an address-range, it chooses the first address in the next address-range for the next microinstruction. After MICRO2 uses the last address in the last range, it uses the address 0000 and issues an error message for each word allocated following the last legal allocation.

2.9.2 Random Allocation

In random mode, constraints can be given to select a set of addresses based on the low order bit configuration. Constraints are described in detail in the next section.

MICRO2 first allocates all absolute assignments and constraints and then allocates the remaining microinstructions starting at the first unallocated address in the first address-range and continuing sequentially through the unallocated addresses of the address space.

2.9.2.1 Constraints - Many microprogrammable microprocessors perform conditional branching by ORing some logic function into the low order bit position of the next microinstruction address. MICRO2 provides a constraint capability for generating a set of addresses for conditional branching.

A constraint consists of an "=" character followed by a constraint string composed of a sequence of 0 and 1 characters.

A constraint specifies a set of addresses. In response to a constraint string, MICRO2 chooses a base address that satisfies the low order bit configuration specified by the constraint. The bits of an address are always ordered from right to left. So the low order bit is the right-most bit.

MICRO2 then assigns the next n microinstructions to the addresses formed by systematically increasing the base address counting only in those bits designated as 0's in the constraint string.

- 2.9.2.2 Indicating a bit that can be 0 or 1 In addition to 0's and 1's, the character '*' can be used in a constraint string. This character informs MICRO2 that it can select an address that has either a 0 or a 1 in that position for the base address.
- 2.9.2.3 The size of the address set The number of microinstructions in the set, n, is determined by the number of zeroes in the constraint string, as follows:

n=2**X

Where X is the number of 0's in the constraint string.

- 2.9.2.4 Constraints Within Constraints If MICRO2 encounters a constraint string within the set of instructions it is allocating to the block of addresses associated with an outer constraint string, it skips to the next address satisfying the inner constraint and then proceeds according to the algorithm specified by the outer constraint. The purpose of the nested constraint is to skip over some addresses that would otherwise be allocated by the outermost constraint.
- 2.9.2.5 Terminating a Constraint A null constraint within the scope of the constraint terminates the constraint. A null constraint is the "=" character. A constraint can also be terminated by an absolute address assignment; however, in this case, MICRO2 issues a warning message.

2.10 COMMUNICATION

In MICRO2, communication among memories in the same program and communication among separate programs can be accomplished.

2.10.1 Memory Communication

Each memory has its own definitions, identification, and address-space. However, if the same field-name is defined in more than one memory, then the value-names defined for that field in any memory are known in all other memories that define the field. This feature permits communication between memories.

2.10.2 Program Communication

Separate programs can be assembled and loaded in a control store by handling address space assignment and communication. If, for example, you wish to have n separate programs, you divide the control store into n+1 logical spaces, namely:

- o Communication Space
- o Space for Program 1
- o Space for Program 2
- o Space for program n

If the address of entry points are fixed, these separate programs can transfer to one another.

2.11 CONDITIONAL ASSEMBLY

The conditional assembly capability permits suppression of the assembly of parts of a program.

2.11.1 The Conditional Assembly Keywords

Three keywords are provided for conditional assembly as follows:

- .IF/expression-name
- .IFNOT/expression-name
- .ENDIF

These keywords divide a program into blocks. The .IF and .IFNOT keywords begin a block. They include an expression-name that is associated with either a true (1) or false (0) value. These keywords have the following meaning.

Keyword

.IF/expression-name If expression-name is associated with a true value (1), assemble the following block; otherwise suppress its assembly.

Meaning

.IFNOT/expression-name If expression-name is associated with a false value (0), assemble the following block; otherwise, suppress its assembly.

In practice, a false value is any value that is not 1. For example, if the expression-name has the value 2, MICRO2 considers it to represent a false value.

2.11.2 Conditional Assembly Blocks

A conditional assembly block begins with either an .IF or .IFNOT and ends with either an .ENDIF for the same expression or another .IF or .IFNOT for the same expression.

2.12 SETTING AND CHANGING EXPRESSION-NAMES

Expression-names are defined and set with the .SET keyword as follows:

.SET/expression-name=expression

Once an expression-name is defined, .CHANGE keyword must be used to change its value.

.CHANGE/expression-name = expression

2.13 LIST CONTROLS

The list controls specify which portions of the output listing are to be produced.

MICRO2 determines whether or not to make a contribution to a file by looking at a counter. If the counter contains a positive number, MICRO2 contributes to the associated file. If the counter is a negative number, MICRO2 does not contribute.

The list controls are as follows:

Keyword	Meaning
.LIST .NOLIST	Increment the listing counter Decrement the listing counter
.CREF	Increment the cross reference counter Decrement the cross reference counter
.BIN .NOBIN	Increment the object counter Decrement the object counter
.EXPAND	List all fields explicitly inserted in an instruction after the last line of the
.NOEXPAND	instruction. Do not list fields.

At the beginning of an assembly, each counter has the value 0.

2.13.1 The List Control Counters

If a list control counter is positive, then MICRO2 creates the specified part of listing. If a list control is negative, MICRO2 suppresses the specified part of the listing.

The counter associated with the .LIST and .NOLIST control determines whether or not an output listing is produced. The counter associated with the .BIN and .NOBIN controls determines whether or not the object part (left field) of the listing is produced. The counter associated with the .CREF and .NOCREF controls whether or not names will be added to the cross reference map. The use of the MICRO2 assembler in the VAX environment is described in the following sections.

2.14 THE VAX 11/780 DEFINITION LANGUAGE

The VAX 11/780 Definition Language describes the VAX 11/780 architecture and provides a macro language for writing microprograms. A listing of this definition is given in Appendix A; the source for the definition language is available on a floppy in the VAX 11/780 WCS kit. This file (VAXDEF.MIC) is copied to SYS\$LIBRARY when the user WCS tools are installed on a system.

You can express the actions of a microprogram in the macros given in the definition language. Then, when you assemble your program, you include the file VAXDEF.MIC as the first input file, as indicated in Appendix A.

2.15 USER INTERFACE

The MICRO2 assembler is called at command level as shown below:

Format

```
MICRO2 input-file-spec

File Qualifiers

LIST[=file-spec]

NOLIST

ULD = [file-spec]

NOULD
```

Prompts

File: input-file-spec

File-Parameters

Input-file-spec

Specifies the names of one or more files to be assembled. If you specify more than one input file, you can use the character '+' to separate file-specs. Input files must not have line numbers; such files are rejected by MICRO2.

Description

MICRO2 assembles the programs contained in the input-file-spec and produces a listing file and an object file.

File Qualifiers

/LIST [=file-spec]

Directs MICRO2 to produce a listing file. If you include a file-spec, MICRO2 uses that file-spec for the listing file. If you do not include a file-spec, MICRO2 uses the name of the input-file, or the name of the first input file in the case of multiple input files, with the default extension .MCR for the listing file.

/NOLIST

Directs MICRO2 to suppress the listing file.

/ULD [=file-spec]

Directs MICRO2 to produce an object-file. If you include a file-spec, MICRO2 uses the file-spec for the output file. If you do not include a file-spec, MICRO2 uses the name of the input, or the name of the first input file in the multiple input file case, with the default extension .ULD for the object file.

/NOULD

Directs MICRO2 to suppress the object file.

Examples

1. MICRO2 ALPHA

MICRO2 assembles the program in the file ALPHA.MIC and produces a listing file ALPHA.MCR and the object file ALPHA.ULD.

2. MICRO2/LIST=BETA ALPHA

MICRO2 assembles the program in the file ALPHA.MIC and produces the listing file BETA.MCR and the object file ALPHA.ULD.

3. MICRO2/NOULD ALPHA+GAMMA

MICRO2 assembles the program formed by the concatenation of ALPHA.MIC and GAMMA.MIC and produces the listing file ALPHA.MCR.

4. MICRO2/LIST=BETA/NOULD ALPHA

MICRO2 assembles the program in the file ALPHA.MIC and produces the listing file BETA.MCR. MICRO2 does not produce an object file because the qualifier /NOULD is given.

5. MICRO2 [SYSLIB] VAXDEF+MYPROG

MICRO2 assembles the definition language in the file SYS\$LIBRARY: VAXDEF.MIC and the microprogram in the file MYPROG.MIC and produces the listing file VAXDEF.MCR and the output file VAXDEF.ULD.

File Specifications

MICRO2 accepts any legal VAX filename. For the purpose of error reporting, MICRO2 abbreviates the filename to the first six characters and the extension to the first three characters.

CHAPTER 3

LOADING A MICROPROGRAM

MICLD is a BLISS program that runs under user control in the VAX/VMS environment. MICLD loads the object files (.ULD) produced by MICRO2 into the Extended Writable Control Store.

MICLD requires kernel (CMKRNL) and error logging (BUGCHK) privileges. It uses VAX privileged registers and instructions to load the microprogram into the WCS and to report the WCS load in the system error log. To use MICLD, therefore, you must have privileges to execute in kernel mode.

3.1 FUNCTIONS

In loading a microprogram, MICLD does the following:

- o Verifies that the Extended WCS board is installed.
- o Initializes each word of the Extended WCS to a special pattern.
- o Loads the set of ULD formatted object modules that make up the microprogram into the Extended WCS.
- Optionally sets the entry vector (VAX address 10E0 hex) to jump to the place in the microprogram where execution begins.
- o Records the fact that the WCS was loaded into the system error log.

The following sections consider each of these functions in detail.

3.1.1 Verifying the Installation of the Extended WCS Board

MICLD first checks that the Extended WCS Board is physically and operationally present in the system. If MICLD finds that the Extended WCS Board is not installed, it issues an error message and exits back to the operating system.

3.1.2 Initializing the Extending WCS

If MICLD finds that the Extended WCS is properly installed, it then initializes each word of the Extended WCS, starting at 1800 and continuing through 1FFF (1C00 through 1FFF if G&H is present) to an initialization pattern.

You can specify the initialization pattern for MICLD to use by including the file qualifier /INITIAL, as described in Section 3.3. If you do not specify an initialization pattern, MICLD uses the default pattern given in Figure 3-1.

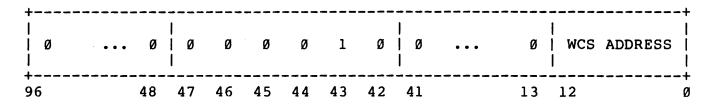


Figure 3-1. Default Initialization Pattern

Bits Ø through 12 of the default initialization pattern contain the address of the microword in the WCS being initialized. Bits <47:42> contain a two (2) to guarantee that no reads or writes will occur to the main memory should the microword inadvertently be executed. All other bits in the 96-bit wide microword are set to Ø.

After execution of MICLD, any microaddress not explicitly loaded contains the initialization pattern. If a microprogram mistakenly jumps to a word that contains the default initialization pattern, then the execution of that word causes a branch to itself with no-op function. The machine then loops at that instruction, continuing to branch to itself. Thus, if your program branches to an unexpected address, you get both protection and information. To recover control from this microcode loop, you enter the console "INIT" command while the console is in console command mode. If the console is not in console command mode, entering Control-P at the console will get it there.

If a microprogram gets into an instruction loop as described above, you must manually reboot the system and then reload the WCS.

3.1.3 Setting the Starting Address

The entry vector (VAX address 10E0 hex) must be set to the address at which the microprogram in the Extended WCS begins. MICLD sets the entry vector to an address if an /ENTRY qualifier is given in the command line. If an address is given with the /ENTRY qualifier, MICLD interprets that address as a hexadecimal number and uses that number to set the entry vector. If an address is not given, then MICLD uses the address of the first microinstruction in the ULD file to set the entry vector.

If an /ENTRY file qualifier is not given, the entry vector must be set by using the console program or the privileged instructions before the execution of the microprogram is attempted. The entry vector is discussed further in section 4.1.2.

3.1.4 Loading the Microprogram

MICLD loads the microwords specified in the ULD file. MICLD begins by creating an image of the loaded WCS. It initializes this image to the initialization pattern and then reads the ULD files, starting with the first word in the first file and continuing sequentially until the last word of the last file.

Each entry in the ULD file for a microword contains both the address of the microword and its contents. MICLD uses the address to determine the position within the image in which the contents of the word is to be stored.

When MICLD finishes reading the last word of the last file, it loads the created image sequentially into the control store.

MICLD permits over-writing. That is, it lets you load an address more than once. When MICLD loads a word into any address, it checks to see if the address has been loaded previously. If so, MICLD issues a warning message and then loads the new word into the given address, destroying the previous contents.

If you take advantage of the over-writing capability of MICLD, you must be careful about the order in which you specify files when you call MICLD. If you do not over-write the WCS, then you can specify your files in any order.

MICLD issues a message at the end of the loading process, indicating whether or not the loading was successful.

3.1.5 Logging the WCS Load

Each time MICLD runs successfully, it makes a note in the system error log that the WCS contents has been changed. These notes can help the system manager determine which user microcode is in the WCS, or relate system problems to particular pieces of user microcode.

3.2 VERIFYING THAT THE LOADING WAS SUCCESSFUL

It sometimes happens that, due to a hardware malfunction, the WCS is not properly loaded and MICLD is not able to detect that fact. The WCS is a write-only memory and MICLD, thus, cannot verify its contents by reading it back after loading and comparing its actual contents with its expected contents.

Erratic or unexpected performance of the executing microprogram can indicate that the WCS is not properly loaded. However, such behavior can also mean that the microprogram is not completely debugged. Under these circumstances, you can try to validate the loading process by one of the following methods.

3.2.1 Using The Sample Program

One way to try to validate the loading process is to load and execute a program whose behavior is known. The sample microprogram given in Appendix B can be used for this purpose. A command file that uses a test program to verify the installation of the tools and the loading process is included in the VAX 11/780 WCS kit.

To invoke this command file in the VMS environment, type:

@[SYSEXE]WCSTOLTST

This command file assembles the sample microprogram (BSERCH) given in Appendix B utilizing the VAX Definition Language given in Appendix A. It then loads the resulting object file into the extended WCS and executes the sample assembly language test program (BSTEST) given in Section B.4. BSTEST executes an XFC causing the loaded sample program to be executed.

After execution of the microprogram, control returns to BSTEST. If the microprogram executed properly, BSTEST prints the following message on the terminal:

"Successful Test Completion"

3.2.2 Sequencing With The Debugger

Another way to validate the loading process is to invoke the WCS debugger from the VAX console and single step the microprogram. You can then observe if the correct sequence of address is executed. The debugger is described in Appendix D of the VAX 11/780 Data Path Description.

Since the WCS is a write-only memory, the debugger is not able to read its contents. You must create a floppy disk image of the contents of the WCS. The debugger then gives the illusion of reading the WCS by reading this floppy disk image that contains the microwords loaded into the WCS. Under normal circumstances, this method of operation is effective. However, the debugger cannot be used to validate the loading process except, as described above, by sequencing through the microprogram.

3.3 USER INTERFACE

To load the Extended Writable Control Store, use the following command:

Format

```
MICLD input-file-spec,...

File Qualifier

-----

/INITIAL=pattern

/ENTRY[=hex-address]
```

Prompts

File: input-file-spec,...

File Parameters

Input-file-spec,...

Specifies the names of one or more files to be loaded into the Extended WCS. If you specify more than one input file, you can use either the character '+' or the character ',' to separate file-specs.

Description

MICLD loads the files you give in the order specified. If you want more than one file to coexist in the WCS, then you separate the filenames with the '+' character.

In the VAX command language syntax, the ',' separator calls for the individual application of the program to each file. Thus, if you use the ',' separator, MICLD initializes the WCS and loads the first file, then initializes the WCS and loads the second file, and so on. The use of the ',' separator has little, if any, legitimate use in the loader command line.

File Qualifiers

/INITIAL=pattern

Specifies the pattern for MICLD to use to initialize the WCS. Pattern is a string of right-justified hexadecimal digits. Any missing digits are padded with zeroes. If you do not give this qualifier, the default pattern given in Section 3.1.2 is used.

/ENTRY[=hex-address]

Specifies the address at which the microprogram in the extended WCS begins. If you do not specify a hex-address, the loader assumes that the microprogram begins at the address of the first word in the first ULD. If you do not give this qualifier, you must set the starting address as described in Section 4.1.2.

Examples

1. MICLD ALPHA+BETA

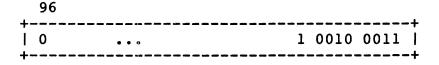
MICLD initializes the WCS to the default pattern and loads ALPHA.ULD and then loads BETA.ULD.

2. MICLD ALPHA, BETA

MICLD initializes the WCS to the default pattern and loads ALPHA.ULD. It then initializes the WCS again and loads BETA.ULD.

3. MICLD/INITIAL=123 ALPHA

MICLD initializes the WCS to the pattern specified, namely:



Then, it loads ALPHA.ULD.

4. MICLD/INITIAL=123/ENTRY=1400 ALPHA

MICLD initializes the WCS to the pattern specified by 123 (as shown above), sets the entry vector to begin execution at address 1400 (hex), and loads the file ALPHA.ULD.

3.4 PROGRAM BEHAVIOR

After you give MICLD the list of files, it loads the files specified into the WCS.

If MICLD detects errors or special circumstances (such as over-loading), it issues a message. At the completion of the loading process, MICLD issues a final message indicating whether or not the loading process was successful.

3.4.1 VAX-11/780 WCS Architecture Description

The Extended WCS is 96 bits wide by 2K occupying VAX addresses 1800 (HEX) through 1FFF. The WCS is divided into three 32 bit X 2K pieces, termed banks. These banks are referenced as BANK 0, BANK 1, and BANK 2 (see Figure 3-2). MICLD loads a microprogram into the Extended WCS one bank at a time. That is, it breaks each microword into three 32-bit pieces and then loads the pieces consecutively into BANK 0, 1, and 2.

Two VAX 11/780 Processor Specific registers support the WCS, namely: the WCS Address Register (WCSA) and the WCS Data Register (WCSD).

The WCSA register consists of 32 bits. The first 16 bits are not used; the last 16 bits are divided into three fields. The WCS ADDRESS field, occupying bits 0 through 12, points to the current WCS address being loaded. The BANK SELECT field (CTR), occupying bits 14 and 13, contains a value of 1, or 2 representing the current bank being loaded. The PIN field, occupying bit 15, is set to 1 if any writes are done with inverted parity. MICLD sets the PIN field to 0. The WCSA register is identified in VAX as processor register number 44.



Figure 3-2 WCSA Register

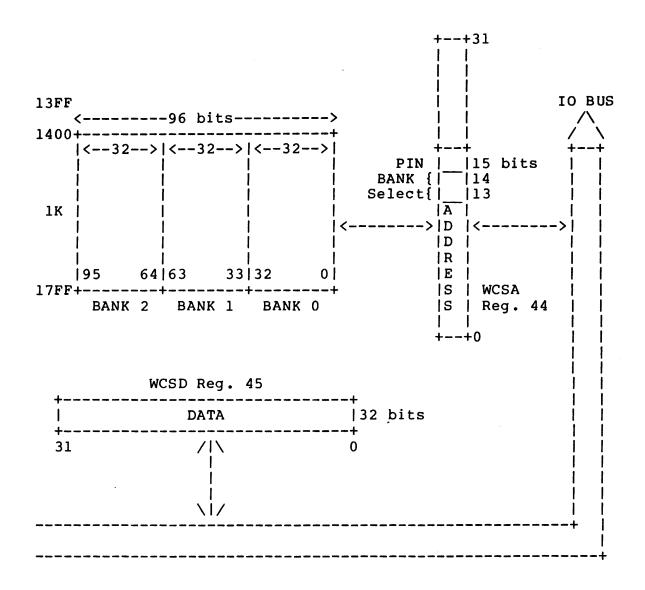


Figure 3-3

The WCSD register consists of 32 bits and contains the microcode or data to be loaded into the WCS. The WCSD register is identified in VAX as processor register number 45.

Information is written into and read from these two VAX processor registers using the Move to Processor Register (MTPR) and Move from Processor Register (MFPR) privileged instructions provided by the VAX-11 system.

To load the user WCS, MICLD must first initialize the WCSA register with the WCS address where the loader is to begin loading microcode. The Bank Select bits are set to 00. Several actions take place to load a complete microword into the WCS.

- Step 1: MICLD loads the first 32 bits of the microword (destined for BANKO) into the WCSD register.
- Step 2: VAX autonomously loads the data in the WCSD register into the User WCS at the location specified by the WCSA register and then auto increments the BANK SELECT field for BANK 1.
- Step 3: MICLD loads the second 32 bits of the microword (destined for BANK 1) into the WCSD register.
- Step 4: VAX repeats Step 2, placing data in BANK 1 at the same WCS location and auto increments the BANK SELECT field for BANK 2.
- Step 5: MICLD loads the third 32 bits of the microword (destined for BANK 2) into the WCSD register, completing the WCS load of the microword.
- Step 6: VAX repeats Step 2, placing the data into BANK 2, resets the BANK SELECT field for BANK 0, and increments the WCS ADDRESS field to point to the next WCS address.

MICLD is then ready to begin loading the next microword into the WCS.

3.4.2 VMS Operating System Support

Programs running under VMS execute in an assigned processor mode. Listed in ascending order of processor privilege and descending access capability, there are four processor modes for programs: 1) user, 2) supervisor, 3) executive, and 4) kernel. All programs begin at user mode and may change processor modes depending on the privileges initially assigned to it.

MICLD begins at user level. MICLD uses the Change Mode to Kernel command (CHMK) to elevate itself to kernel mode. The CHMK command is documented in the VAX 11/780 Architecture Handbook, Volume 1, Section 13-2.

Successful execution of the CHMK command and other processor mode commands depends on the program being run in an account equipped with the necessary privileges.

3.5 ERROR MESSAGES

MICLD issues error messages in the VAX standard error message format. Namely:

%cprogram-name>-<severity-code>-<abbreviation>,<message>

Where:

<abbreviation> is a short identifier for the message
<message> is the error message text

The abbreviations and messages produced by MICLD are as follows:

Abbreviation	Message
ABKEYW	ambiguous keyword
AMVERB	ambiguous verb
BADENTRY	/ENTRY value not a hex number
BADENTRY	/ENTRY value not in the user WCS address range
BADINIT	initial value conversion error
BADULD	data record conversion error
BADULD	invalid data record syntax
BADULD	missing equal sign
BADULD	missing right bracket
CHALOG	unable to log WCS content change
FNF	file not found <filename></filename>
INVCMD	invalid command format
NOPARM	missing parameter
NOPRIV	no kernel mode privileges
NOPRIV	unable to write to error log
NOWCS	WCS memory not installed
TERMEOF	end of file on terminal

WCSBND address out of bounds at <address>(hex)

WCSLOAD WCSA error <address> should be <address> (hex)

WCSCHANGE WCS content changed by <userid> using file

<filename>

WCSMLO memory location overwritten at <address>(hex)

CHAPTER 4

EXECUTING A MICROPROGRAM

To execute a microprogram in the extended WCS, the following actions are necessary:

- 1. An XFC instruction in a main memory program must be executed.
- 2. The field consisting of Bits 1 and 0 of Vector 14 of the System Control Block must be set to 2.
- 3. The entry vectory (10E0 hex) must be set to jump to the first instruction to be executed in the microprogram in the Extended WCS.

The following sections describe these actions in detail.

4.1 EXTENDED FUNCTION CALL

A microprogram is invoked by the XFC instruction. The Extended Function Call (XFC) instruction provides a controlled mechanism for software to request services of non-standard microcode in the extended Writeable Control Store (WCS). The request is controlled by the system control block. All opcodes reserved to the extended WCS start with FC (hex), which is the XFC instruction, using the format:

FC

The XFC instruction has no parameters.

You can pass parameters either as normal operands or in fixed registers. For example, if you have more than one extended function resident in the WCS, you can use the bytes following the opcode to specify which of several extended functions are requested.

Execution of the XFC instruction generates what is called an exception.

4.1.1 Exceptions

The notification of an event relevant primarily to the currently executing process, which invokes software in the context of the current process, is termed an exception.

Exceptions are handled by, or trapped to, operating system software. Further, all exceptions either wait for the instruction that caused them to complete before happening or they restore the processor to the state it was in prior to executing the instruction that caused the execution.

An exception caused by the execution of an XFC instruction (classified as a fault) occurs during an instruction, leaving the registers and memory in a consistent state such that elimination of the fault condition and restarting the instruction will give correct results. The XFC instruction causes faults called the opcode reserved for customer and customer reserved exceptions. The value of the PC that is saved on the stack points to the instruction faulting.

Exceptions are usually reflected to the originating mode as a signal. In general, the signal is interpreted via a vector in the system control block. Separate vectors are defined for each class of exception and interrupting device controller.

4.1.2 <u>Setting The System Control Block Vector</u>

When an XFC is executed, VMS handles the fault by trapping to vector 14 (hex) of the system control block (SCB). VMS examines the low order two bits of the vector and if it finds the value 2, then it traps to the entry vector (10E0 hex).

The two low order bits of Vector 14 of the SCB must be set to 2 to execute a microprogram in the Extended WCS. The console data deposit command can be used to set vector 14 relative to the base SCBB.

4.1.3 Patching The Entry Vector

After vector 14 is accessed, the system traps to address 10EO (hex), which is resident in the VAX microcode area and is called the entry vector. The entry vector must be loaded with a JUMP microinstruction to the desired entry point in the extended WCS. This extended WCS entry point is usually a control routine or exception handler; however, it can simple be the first instruction in the microcode function to be performed.

The entry vector is set during the loading process by MICLD if an /ENTRY file qualifier is given in the MICLD command line. The entry vector can also be set at the VAX console using the console program.

Note, that when the fault occurs, the system traps (in the end) to the user WCS. The user defines an exception handler in microcode to service the "extended customer opcode" fault.

The WCS contains only one application, there is no need for a handler to resolve what function should be performed in the WCS. However, if the WCS contain several microroutines, an exception handler must resolve event by accessing additional data. The microprogrammer must have a good understanding of the VAX micro machine data path to develop this exception handler.

More information on the System Control Block and the handling of exceptions can be found in the VAX 11/780 Hardware Handbook.

APPENDIX A

VAX 11/780 FIELD AND MACRO DEFINITIONS

The VAX 11/780 Definition Language identifies the fields of the microword and provides a macro language to aid you in writing microprograms. The sample microprogram in Appendix B is written in this definition language.

When assembling a program written in this definition language, you include the definition language source by concatenating your program file with the definition file VAXDEF.MIC in the MICRO2 command line as follows:

\$ MICRO2 [SYSLIB] VAXDEF+MYPROG

In the above example, MYPROG.MIC is the name of the file that contains the source MICRO2 microprogram.

The definition language file is available on a floppy in the VAX 11/780 WCS kit. It is copied to SYS\$LIBRARY:VAXDEF.MIC when the kit is installed on a system.

IN L ADEC REF /NAT				.		a. 1	#R	AD:	T I S NI IX : PRES	JMBI 16 35 (ERE) OREI	D . F .	FRO ON	M (0 O	N T ERE	HE ST	R IN	IGI G F	HT FIE	OF ELD:		FI	ELD									h an r-			_4_	
]	гвс	,	:	DK	:	SH	F	Bi	1X	AM		D T			BE	:N		A		1	ALL	J	SU	B:		ĸ	мx		:	SI			QK		:	s
: 5	5 4	3	2	1 1	0 '	 3 8: 9 8: +	76	5	4 3	3 2	1 (01	 7 7 9 8	 7 7	7 7	5 4	3	2	11	0:	9 1	3 7	7 6	15	4 : :	3 :	? 1	0	9	813						- -	 5 0
,														R B					•					•	1					1	AC	м				-1-	
														X 	1										Τ.	,							•				
+	:			1:	-+- F: B:		SPI	-	v eta un a	-+ : : : : :				+	! 	-	IA IK	IF IE IK	K	C		:	M X	ISM I	-+· 	E A		+	-							• •• •	
A D S +	+-4	- 4 5	4	4	F 	1 4 5	3 3	 3 7	3 3 6 5	; ; ; ; ; ; ; ;	3 3 3	-+- 313 213	IEK 3 3	+	MS0	2 2	V A K	F K - 2 4	1C: 1K: 1 : 1-1 12:	2	2 2	-+- 2:1	B M X 	SM 1 7	11:	1 1 5 4	1 3	12	1	0 9	8 9	0 7	0	5 -	4 3	3 2	2
A D S +	+-4	5 -	 4 4 	4	F 	. 0 9 S P	3 3 9 8	3 7 +	3 3 6 5 6 5	1314	3 3 3	-+- 313 213	IEK 3 3	+	MS0	2 2	V A K	F K - 2 4	1C: 1K: 1 : 1-1 12:	2	2 2	-+- 2:1	B M X 	SM 1 7	11:	1 1 5 4	1 3	12	1	0 9	8 9	0 7	0	5 -	4 3	3 2	2
A D S +	+-4	5 -	4	4 3	F 	S F O A	3 3 3 9 8	3 7 + 1 S 1 A	3 3 6 5 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6	; 4 ; 4 ; 4	3 3 3	-+- 313 213	IEK 3 3	+	MS0	2 2	V A K	F K - 2 4	1C: 1K: 1 : 1-1 12:	2	2 2	-+- 2:1	B M X 	SM 1 7	11:	1 1 5 4	1 3	12	1	0 9	8 9	0 7	0	5 -	4 3	3 2	2
A D S +	+-4	5 -	 CT 	4 3	F 	S P O · A C	3 3 3 9 8	3 7 + S +	3 3 3 6 5 5 6 5 6 5 6 6 6 6 6 6 6 6 6 6	1	3 3 3	-+- 313 213	IEK 3 3	+	MS0	2 2	V A K	F K - 2 4	1C: 1K: 1 : 1-1 12:	2	2 2	-+- 2:1	B M X 	SM 1 7	11:	1 1 5 4	1 3	12	1	0 9	8 9	0 7	0	5 -	4 3	3 2	2

LA=0 RAMX=1

RAMX.SXT=2 RAMX.OXT=3

: ACF, ACM, ADS, ALU, AMX* . TOC *Machine definition ACF/=<71:70>,.DEFAULT=0 **FACCELERATOR CONTROL** NOP=0 SYNC=1 TRAP=2 CONTROL=3 FACCELLERATOR-DEPENDENT CONTROL FUNCTION JACCELERATOR MISCELLANEOUS CONTROL ACM/=<57:55> PWR.UP=0 ABORT=1 FRETURN ACCEL TO MONITORING IRD POLY.DONE=6 ADS/=<47:47> #ADDRESS SELECT VA≔0 IBA=1 **FALU CONTROL FUNCTIONS** ALU/=<69:66>,.DEFAULT=OF A-B=00 A-B.RLOG=01 A-B-1=02 INST.DEF=03 A+B+1=04 #INSTRUCTION DEPENDENT A+B=05 A+B.RLOG=06 ORNOT=07 XOR=08 #A .OR. .NOT. B
#A .XOR. B
#A .AND. .NOT. B
#.NOT. A ANDNOT=09 NOTA=OA A+B+PSL.C=OB #A .OR. B OR≖OC AND=OD B=0E A=OF AMX/=<81:80> JAMX TO ALU

FRAMX SIGN EXTENDED ACCORDING TO DT

FRAMX ZERO EXTENDED. OXT(L)=0

```
: BEN, BMX"
               *Machine definition
 . TOC
BEN/=<76:72>,.DEFAULT=0
                                                          FBRANCH ENABLE
              NOP=0
                                                          INO BRANCH
                                                           # ALU Z
              Z=1
                                                          #LA<1>, FSL<C>, LA<0>
              ROR≈2
                                                          # ALU C31, O
#OUTPUT OF EXTENDED IRC-ROM
              C31 = 3
               IRC.ROM=4
                                                          #IB O READY ?
#CODE FROM ACCELERATOR
#(VAX MODE) *, ASRC+VSRC, ASRC+Q+D
               IB.0=5
               ACCEL=6
               DATA.TYPE=8
                                                        ; (VAX MODE) *, ASKC+VSKC, ASKC+VFD;
; O--NORMAL, 1--QUAD OR DOUBLE;
; 2--FIELD, 3--ADDRESS;
;(-11 MODE) *, O CLASS, J CLASS+DM27;
;(VAX MODE) *, IR<2:1>
;(-11 MODE) *, SM47+SM57+DM47+DM57, DST R=PC;
;(VAX MODE) MODE.LSS.ASTLVL, *, *
               END.DP1=8
               IR2-1=9
               FC.MODES=9
              REI=OA
                                                        ;(-11 MODE) SRC R=PC
; 0--TB MISS, 1--ERROR
; 2--STALL, 3--DATA OK
;SC.NE.O, D<1:0>
              SRC.FC=0A
               IB.TEST=OB
              MUL = OC
                                                         ;SC.NE.O, D<1:0>
;Q<31>, D.NE.O, D<31>
;AC LOW, INTERNAL INTERRUPT, INT REQ
;O, D BYTE O VALID DIGIT, D2-0 NEG SIGN
;MICROTRAP DISPATCH VECTOR
;-FPD, NESTED ERROR, LOW TWO BITS:
; O--READ INTERLOCK, 1--READ, READ CHK
; 2--WRITE, 3--READ, WRITE CHK
               SIGNS=OD
               INTERRUPT=0E
               DECIMAL=OF
               UTRAP=10
              LAST.REF=11
                                                         #EALU N, EALU Z, SC.NEQ.O, SS
#SC<9:8>.NE.O, SC.GT.O, SC<9:5>.NE.O
              EALU=12
              SC=14
                                                         FRLOG EMPTY, ALU<1:0>=0, ALU<1>, ALU<0>
; (ALU BITS FROM PREVIOUS STATE)

FSTATE <7:4>
FSTATE <3:0>
FBYTES 3, 2, 1, 0 OF D.NE.0
              ALU1-0=15
              STATE7-4=16
              STATE3-0=17
              D.BYTES=18
                                                          #D<3:0>
              D3-0=19
              FSL.CC=1A
                                                          #N,Z,V,C OF PSL
                                                          ;ALU N, ALU Z, IR<0>, ALU C31
;-VA<31:30>, -CONSOLE, IS+CM, KERNEL
              ALU=1B
              PSL.MODE=1C
                                                          #FTE VALID, ALIGNED, QUAD, +
### 0--TRANSLATION OK, 1--WR CHK AND M=0
#### 2--ACCESS VIOLATION, 3--TB MISS
               TB.TEST=1D
                                                          FBMX TO ALU
BMX/=<84:82>
                                                          #A O IN THE BIT SELECTED BY SC<4:0>
              MASK=0
                                                          JLB UNLESS R=PC, THEN PC
              PC.OR.LB=1
                                                          FRACKED FLOATING
              PACKED.FL=2
              LB=3
              LC≈4
              PC=5
               кмх≖6
               RBMX=7
                                                          FD OR Q
```

. TOC *Machine definition : CCK, CID, DK, DT'

CCK/=<22:20>,.DEFAULT=0 #CONDITION CODES

	*Note : * =	RESERVED OPERATION	1, "AL	.U S	IGN' AND 'A	MX SIGN" ARE SIZE	DEPEND	ENT .
	,	NATIVE MODE PSL			C	OMPATIBILITY MODE	FSL	
	N	Z	V	С	N	Z	V	C
NOP=0	N	ż Z	V	Ç	N.	ž Z	l v	<u> </u>
LOAD.UBCC=1 SET.V=2 ,.VALIDITY= <v1:< td=""><td>F N</td><td>;</td><td>1 1 1</td><td>C</td><td>; N</td><td>;</td><td>: • :</td><td>* :</td></v1:<>	F N	;	1 1 1	C	; N	;	: • :	* :
N_AMX.Z_TST.VC_VC=3 ROR=4VALIDITY= <vo< td=""><td># AMX SIGN</td><td>: Z.and.(ALU.eq.0)</td><td>1 1 1</td><td></td><td>AMX SIGN</td><td>: Z.and.(ALU.eq.0) : ALU.eq.0</td><td>: V :</td><td>C :</td></vo<>	# AMX SIGN	: Z.and.(ALU.eq.0)	1 1 1		AMX SIGN	: Z.and.(ALU.eq.0) : ALU.eq.0	: V :	C :
NZ_ALU.VC_0=5	ALU SIGN		10	-	ALU SIGN		0	0
NZ_ALU.VC_VC=6 ,.VALIDITY= <v13 C_AMX0=6 ,.VALIDITY=<v03< td=""><td></td><td>ALU.ea.0</td><td>1</td><td></td><td>l N</td><td>i z</td><td> v </td><td>AMX<o> i</o></td></v03<></v13 		ALU.ea.0	1		l N	i z	v	AMX <o> i</o>
INST.DEP=7	; ;	+	struc -+	tio	n derendent +	+	-++	:

CONSOLE & ID BUS CONTROL IF FS/1
DEFAULT, ALLOW AUTO IB READ
SET CONSOLE ACKNOWLEGE FLAG
CLEAR CONSOLE MODE
FREAD ID BUS REG SELECTED BY SC
FREAD ID BUS REG SELECTED BY UKMX
WRITE REG SELECTED BY UKMX
WRITE REG SELECTED BY UKMX CID/=<45:42> NOP=1 ACK=5 CONT=7 READ.SC=9 READ.KMX=0B WRITE.SC=OD WRITE.KMX=OF

DK/=<91:88>,.DEFAULT=0 NOP=0 LEFT2=1 RIGHT2=2 DIV=4

> LEFT=5 RIGHT=6 SHF.FL=9 ACCEL=OA BYTE.SWAP=OB Q≔OC DAL.SC=OD DAL.SV=OE

CLR=OF DT/=<79:78>,.DEFAULT=0

> LONG=0 WORD=1 BYTE=2 INST.DEP=3

DEFAULT, HOLD
DOUBLE SHIFT LEFT
DOUBLE SHIFT RIGHT
FOR ALU CRY, SHIFT LEFT
FELSE LOAD FROM SHF
SHIFT LEFT
SHIFT RIGHT
LOAD SHF MUX, INTEGER FORMAT
LOAD SHF MUX, UNPACKED FLOATING FORMAT
LOAD ACCELERATOR DATA FROM DF BUS
REFLECT BYTES AROUND BIT 16 FREFLECT BYTES AROU
FLOAD Q THRU DAL
FLOAD DAL[SC]
FLOAD DAL[SHF VAL]
FLOAD ZEROS

#DATA TYPE CONTROLS AMX SIGN/ZERO EXTENDER, SHF AMOUNT, CONDITION CODE SETTING, AND MEMORY REFERENCES DEFAULT

#INSTRUCTION DEPENDENT -- #ANY OF ABOVE, OR QUAD/DOUBLE

```
: EALU, EBMX, FEK, FS, IEK, IBC*
. TOC
            *Machine definition
                                                JEXPONENT ALU
EALU/=<15:13>
            A=0
OR=1
            ANDNOT=2
            B=3
            A+B=4
            A-B=5
            A+1=6
            NABS.A-B=7
                                                #-ABS(A-B)
EBMX/=<19:18>
                                                FERMX TO EALU
                                                FDEFAULT
            FE=0
            KMX=1
            AMX.EXP=2
            SHF.VAL=3
                                                #SHIFT VALUE
FEK/=<24:24>,.DEFAULT=0
NOF=0
                                                FFE REGISTER CONTROL
                                                DEFAULT, HOLD
            LOAD=1
FS/=<42:42>
                                                FUNCTION SELECT FOR 43-46
            MCT=0
                                                FENABLE MEMORY CONTROL
            CID=1
                                                FENABLE ID BUS AND CONSOLE CONTROL
                                                FINTERRUPT AND EXCEPTION ACKNOWLEDGE
IEK/=<31:30>
            NOP=0
            ISTR=1
                                                *STROBE INTERRUPT REQUESTS
                                                FINTERRUPT ACKNOWLEDGE FEXCEPTION ACKNOWLEDGE
            IACK=2
           EACK=3
IBC/=<95:92>,.DEFAULT=0
                                                FIRUF CONTROL FUNCTIONS
           NOP=0
STOP=1
                                                *DEFAULT
           FLUSH=2
                                                FRLUSH IB AND LOAD IBA
           START=3
CLR.0.1=4
                                               ;CLEAR BYTES 0,1 (-11 OPCODE)
;CLEAR BYTES 2,3 (-11 ISTREAM DATA)
;TRANSFER BRANCH DISPLACEMENT
            CLR.2.3=5
            BDEST=7
                                               #CLEAR BYTE 0 (VAX OPCODE)
#CLEAR BYTE 1 (VAX SPECIFIER)
            CLR.O=OC
            CLR.1=0D
                                               CLEAR BYTE 1 (VAX SPECIFIER)
CLEAR BYTES 0-3 (-11 OP & DATA)
CLEAR BYTES 1-5 CONDITIONALLY
IF THERE IS NO SPECIFIER EVALUATION,
CLEAR NOTHING. IF A SELF-CONTAINED
SPECIFIER, CLEAR IT. IF IMMEDIATE,
ABSOLUTE, OR DISPLACEMENT, CLEAR THE
ISTREAM LITERAL.
           CLR.0-3=0E
CLR.1-5.COND=OF
```

.TOC	*Machine definition	: ID.ADD	R, J"
ID.ADDR	/=<63:58>	FID BUS	ADDRESS
	IBUF=0	#RD	#SPECIFIER/LITERAL DATA FROM IB
	DAY.TIME=1	#RD+WR	CURRENT TIME OF DAY
			# MUST READ UNTIL STOPS CHANGING
	SYS.ID=3	FRD	SYSTEM IDENTIFICATION
	RXCS=4	#RD+WR	
	RXDB=5	#RD	CONSOLE RECIEVE DATA BUFFER (TO-ID REGISTER)
	TXCS=6	#RD+WR	CONSOLE TRANSMIT CONTROL/STATUS
	TXDB=7	∌ WR	CONSOLE TRANSMIT DATA BUFFER
			(FROM-ID REGISTER)
ŷ	DQ=8		#DATA PATH D/Q REGISTERS (MAINT ONLY)
	NXT.PER=9	# WR	FINTERVAL CLOCK NEXT PERIOD REGISTER
		#RD+WR	#INTERVAL CLOCK CONTROL/STATUS
	INTERVAL=OB	#RD	CURRENT INTERVAL COUNT
	CES=0C	#RD+WR	#CPU ERROR/STATUS
	VECTOR=OD		FEXCEPTION & INTERRUPT CONTROL
	SIR=OE	#RD+WR	
	PSL=OF	#RD+WR	
	TBUF=10		TRANSLATION BUFFER DATA
	TBERO=12		FTB ERROR/STATUS O
	TBER1=13		*TB ERROR/STATUS 1
	ACC - 0=14		#ACCELERATOR REGISTER #0
	ACC - 1 = 15		#ACCELERATOR REGISTER #1
	ACC.2=16		JACCELERATOR REGISTER #2
	ACC.CS=17		∮ACCELERATOR CONTROL/STATUS ∮NEXT ITEM FROM SBI HISTORY
	SIL0=18		
	SBI.ERR=19		SBI ERROR REGISTER
	TIME.ADDR=1A		#SBI TIMEOUT ADDRESS #FAULT/STATUS
	FAULT=1B		#SBI SILO COMPARATOR
	COMP=1C		#SBI MAINTENANCE
	MAINT=1D PARITY=1E		#CACHE PARITY
	USTACK=20		#MICROSTACK
	UBREAK=21		#MICRO BREAK
	WCS.ADDR=22		FILESCING ACIDETIS
	WCS.DATA=23	# WR	*WRITING WCS COUNTS ADDRESS

FID BUS ADDRESSES CONTINUED. ADDRESSES 24-3F ARE RAM LOCATIONS

POBR=24 P1BR=25 SBR=26 KSP=28 ESP=29 SSP=2A USP=2B ISP=2C FPBA=2D D.SV=2E Q.SV=2F	PROCESS SPACE O BASE REGISTER PROCESS SPACE 1 BASE REGISTER SYSTEM SPACE BASE REGISTER PROCEDITOR P
T0=30 T1=31 T2=32 T3=33 T4=34 T5=35 T6=36 T7=37 T8=38 T9=39	;GENERAL TEMPS
PCBB=3A SCBB=3B POLR=3C P1LR=3D SLR=3E	#PROCESS CONTROL BLOCK BASE #SYSTEM CONTROL BLOCK BASE #PROCESS O LENGTH REGISTER #PROCESS 1 LENGTH REGISTER #SYSTEM LENGTH REGISTER

.CREF

J/=<12:0>,.NEXTADDRESS

*NEXT MICRO WORD ADDRESS

.NOCREF

```
. TOC
          *Machine definition
                                          : KMX*
KMX/=<63:58>
                                FCONSTANTS OR # FROM FK
                                ##8 FROM FK
          .8=0
          .1=1
                               ##1 FROM FK
##2 FROM FK
                                ##3 FROM FK
##4 FROM FK
           .3=3
           . 4=4
          SP1.CON=5
SP2.CON=6
                                #SPECIFIER 1 CONSTANT
#SECIFIER 2 CONSTANT (-11 MODE)
# OR ZEROS (VAX MODE)
          ZERO=6
                                #SCE9:03 FROM FK
          SC=7
#8 - 3F: CONSTANTS (1 CYCLE SETUP IF ALU IN ARITH MODE)
#DECIMAL VALUE OF CONSTANT
          .14=8
.A0=9
                                #20
#160
           .34=0A
                                152
          .28=0B
.40=0C
                                $40
$64
           .50=0D
                                ;******* .3000=0E If system rev is less than 6 ******
;239
           .7FF0=0E
.EF=0F
           .80=10
                                1128
           .8000=11
                                )-32768
)255
           .FF=12
           .FF00=13
                                1-256
                                #30
          .1E=14
           .3F=15
                                163
          .7F=16
                                1127
          .7=17
.F=18
                                ;7
;15
           .10≔19
                                116
                                9-24
9-16
          .FFE8=1A
                                #-8
#32
#48
           .FFF8=1C
          .20=1D
.30=1E
          .18=1F
                                #24
           .3FF=20
                                11023
           .C=21
                                #12
           .D=22
                                113
          .1F=23
.1F00=24
                                #31
#7936
          .B0=25
                                9176
          .E003=26
          .7C=27
                                1124
          .FFE0=28
                                1-32
           .60=29
                                ;96
          SPARE=2A
          .DFCF=2B
                                97
          .4000=2C
.FFF1=2D
                                \mathfrak{p}******* .FFEF=2C If system rev is less than 6 ******* \mathfrak{p}=15
          .19=2E
                                125
          •FFF9=2F
                                9-7
```

KMX DEFINITION CONTINUED

.FFFF=30	9 - 1
.88=31	11
.3030=32	\$?
.F0=33	1240
.C0=34	#192
.6=35	\$6
.9=36	19
•FFF6=37	1-10
.FFF5=38	\$-11
.1A=39	#26
.24=3A	136
.1B=3B	127
•FFFC=3C	j - 4
.A=3D	110
.7E=3E	1126
SPARE=3F	

"Machine definition : MCT, MSC" MCT/=<47:42>,.DEFAULT=3E FMEMORY CONTROL FIEST TBUF WITH READ CHECK FNEITHER CPU NOR IB GETS MEM CYCLE TEST.RCHK=00 MEM.NOP=02 THE TRUE WITH WRITE CHECK THE TRUE WITH WRITE CHECK WRITE, INHIBIT TRAPS WRITE, NORMAL VARIETY THE TRUE WRITE, VIRTUAL ADDRESS TREAD, NORMAL VARIETY TREAD, INHIBIT TRAPS TREAD FOR MODIFY THE TRAPS TEST.WCHK=04 WRITE.V.NOCHK=OA WRITE.V.WCHK=OC LOCKWRITE.V.XCHK=0E READ.V.RCHK=10 READ.V.NOCHK=12 READ. V. WCHK=14 FREAD, CHECK CONTROLLED BY IBUFFER BEGIN NEW INSTRUCTION STREAM READ. V. IBCHK=16 READ. V. NEWPC=18 DATA GOES TO INSTRUCTION BUFFER FINTERLOCK READ, INHIBIT CHECK FINTERLOCK READ, NORMAL VARIETY LOCKREAD.V.NOCHK=1A LOCKREAD.V.WCHK=1C #STOP ALL SBI ACTIVITY #RESET SBI #CLEAR CACHE ENTRIES SBI.HOLD=20 SBI.HOLD+UNJAM=22 INVALIDATE=24 VALIDATE=26 *MICRODIAGNOSTIC FORCE VALID #EXTENDED WRITE TO CLEAR MOS ERRORS #WRITE, PHYSICAL #INTERLOCK WRITE, PHYSICAL EXTWRITE.P=28 WRITE.P=2A LOCKWRITE.P=2E FREAD, PHYSICAL FINTERRUPT SUMMARY READ FINTERLOCK READ, PHYSICAL FGIVE IB A CYCLE IF IT WANTS ONE READ.P=32 READ.INT.SUM=36 LOCKREAD.P=3A ALLOW.IB.READ=3E MSC/=<29:26>,.DEFAULT=0 NOP=0 **PDEFAULT** FCREATE NEW PSL FOR CHM

NOP=0 CHK.CHM=01 CHK.FLT.OPR=02 CHK.ODD.ADDR=03 IRD=04 LOAD.STATE=05 LOAD.ACC.CC=06 READ.RLOG=07 CLR.FPD=08 SET.FPD=09 CLR.NEST.ERR=0A SET.NEST.ERR=0A SETOND.REF=0C RETRY.NO.TRAP=0D

RETRY.TRAP=OE INH.CM.ADDR=OF #UTRAP IF ALU<15>=1, ALU<14:7>=0

#THIS STATE IS INSTRUCTION DECODE

#TAKE CONDITION CODES FROM ACCELERATOR
#(AND POP RLOG STACK)
#CLEAR PSL<FPD> BIT
#SET SAME
#CLR NESTED ERROR FLAG IN CPU STATUS
#SET SAME
#OF UNALIGNED DATA REFERENCE
#APPLY SAVED CONTEXT, INHIBIT TRAPS
#APPLY SAVED CONTEXT TO THIS REF
#ALLOW USE OF FULL 32-BIT ADDRESS

.TOC	*Machine definition	: PCK, QK, RAMX, RBMX*
PCK/=<3	4:32>,.DEFAULT=0	#ADDRESS COUNT CONTROL
	NOP=0	# DEFAULT
	PC_VA=1	
	PC_IBA=2	
	VA+4=3	JUA_UA+4
	PC+1=4	\$PC_PC+1
	PC+2=5	#PC_PC+2
	PC+4=6	#PC_PC+4
	PC+N=7	#PC_PC+N, N IS DETERMINED BY INSTR BUFFER
QK/=<54	:51>,.DEFAULT=0	
	NOP=0	DEFAULT, HOLD
	LEFT2=1	DOUBLE SHIFT LEFT 2
	RIGHT2=2	∮DOUBLE SHIFT RIGHT 2
	LEFT=5	
	RIGHT=6	
	SHF=8	ILOAD SHF, INTEGER FORMAT
	SHF.FL=9	FLOAD SHE, UNPACKED FLOATING FORMAT
	DEC.CON=OA	DECIMAL CONSTANT = 6'S IN EACH NIBBLE
		FOR WHICH ALU CRY OUT IS FALSE
	ACCEL=OB	*LOAD ACCELERATOR DATA FROM DF BUS
	D=OC	
	ID=0E	#LOAD ID BUS
	CLR=OF	#LOAD ZERO
RANX/=<	77:77>,.DEFAULT=0	DATA PATH MIXER TO AMX
	D=O	# DEFAULT
	Q=1	
RBMX/=<	77:77>	DATA PATH MIXER TO BMX. SAME BIT AS RAMX
	Q=0	
	D=1	

```
. TOC
           *Machine definition
                                         : SCK, SGN, SHF, SI, SMX*
SCK/=<23:23>,.DEFAULT=0
                                           #SC REGISTER CONTROL
                                           #DEFAULT, HOLD
#LOAD SMX<09:00>
          NOP=0
          LOAD=1
SGN/=<50:48>,.DEFAULT=0
                                           FSIGN CONTROLS
          NOP=0
                                           #DEFAULT
          LOAD.SS=1
SS.FROM.SD=2
                                           #SS_ALU<15>
#SS_SD
#SD_NOT SD
          NOT.SD=3
          SD.FROM.SS=4
                                           #SD_SS
                                           | SBLALU<15>, SSLSS.XOR.ALU<15>
| SBLALU<15>, SSLSS.XOR.ALU<15>.XOR.IR<1>
          SS.XOR.ALU=5
          ADD.SUB=6
          CLR.SD+SS=7
                                           #CLEAR BOTH
SHF/=<87:85>,.DEFAULT=0
                                           FALU SHIFTER CONTROLS
                                           #ALU SHIFIER CONTROLS
#DEFAULT, SHF_ALU
#SHF_ALU(L1), INSERT SI CNTL
#SHF_ALU(R1), INSERT SI CNTL
#SHF_ALU(DT: L0,L1,L2,L3), INSERT 0
          ALU=0
          LEFT=1
          RIGHT=2
          ALU.DT=3
          RIGHT2=4
                                           #SHF_ALU(R2), INSERT SI CNTL
#SHF_ALU(L3)
          LEFT3=5
SI/=<57:55>,.DEFAULT=3
                                           #SHIFT INPUT CONTROLS
                                                     SHF
                                                                D
                                                                           Q
                                                      -----
                                                     PSL<N>
          DIVD=0
                                                                Q31
                                                                           ALU C31
Q31
                                                     ALU 31
          ASHR=1
                                                                QO
          ASHL=2
                                                      ٥
                                                                0
                                                                           D31
           ZERO=3
                                                      0
                                                                0
                                                                           0
           SPARE=4
          DIV=5
                                                     Q31
                                                                Q31
                                                                           ALU C31
                                                                ALU 0,1 0
ALU 0,1 1
          MUL+=6
MUL-=7
                                                      0
                                           #MIXER TO SC
#EALU <9:0>
SMX/=<17:16>
          EALU=0
          FE=1
                                           #FE<9:0>
                                           #ALU<09:00>
          ALU.EXP=3
                                           #ALU<14:07>
```

```
. TOC
                                      : SPO, SPO.AC, SPO.ACN, SPO.ACN11, SPO.R*
          *Machine definition
SPO/=<41:35>,.DEFAULT=0
                                                 *SCRATCH PAD OPCODE, 7 BITS
                                                 #DEFAULT
         NOP=0
          LOAD.LC.SC=6
                                                 $LOAD LC, ADR=SC[03:00]
          WRITE.RC.SC=7
                                                 #WRITE RC, ADR=SC[03:00]
                                                 #4 FUNCTION BITS OF SPO FIELD #LOAD LA, LB FROM R(ACN) #LOAD LA_RN, HOLD LB #WRITE RA, RB (ACN)
SPO.AC/=<41:38>
         LOAD.LAB=1
         LOAD.LA=2
WRITE.RAB=3
SP0.ACN/=<37:35>
                                       FAC NUMBER IN SPO FIELD
                                       JVAX MODE
                                                           RA
                                                                               RB
                                                           SP1 R
          SP1.SP1=0
                                       ŧ0
                                                                               SP1 R
          SP2.SP2=1
                                       ; 1
                                                           SP2 R
                                                                               SP2 R
         SP2.SP1=2
                                       12
                                                           SP2 R
                                                                               SP1 R
         PRN=3
                                       ;3
                                                           PRN
                                                                               PRN
         PRN+1=4
                                                           PRN+1
                                                                               PRN+1
                                                           SC<03:00>
                                                                               SC<03:00>
         SC=5
                                       15
                                                 SP1 R+1
                                                                     SP1 R+1
         SP1+1=6
                                       $6
SPO.ACN11/=<37:35>
                                       JAC NUMBER IN SPO FIELD -- 11 MODE
                                       -11 MODE
                                                              RA
         SRC.SRC=0
                                            0
                                                           SRC R
                                                                               SRC R
DST R
         DST.DST=1
                                                           DST R
         DST.SRC=2
                                            2
                                                           DST R
                                                                               SRC R
                                                                               SRC R
SRC R .OR. 1
         SRC.SRC=3
SRC.OR.1=4
                                                           SRC R
SRC R .OR. 1
                                            3
                                            4
         SC=5
                                            5
                                                           SC<03:00>
                                                                               SC<03:00>
                                       .
                                       #SCRATCH PAD FUNCS WITH LOW 4 BITS OF SP AS ADR
SPO.R/=<41:39>
         LOAD.LC=2
                                       #LOAD LC, ADR=SPO.RN
                                       #WRITE RC
         WRITE.RC=3
                                       #WRITE RC
#LOAD LA, LB
#WRITE RA, RB
#LOAD LA, LB[R1], AND WRITE RC[RN]
#LOAD LC[RN], AND WRITE RA, RB[R1]
         LOAD.LAB=4
         WRITE.RAB=5
         LOAD.LAB1.WRITE.RC=6
         LOAD.LC.WRITE.RAB1=7
```

```
: SPO.RAB, SPO.RC, SUB, VAK*
. TOC
           "Machine definition
SPO.RAB/=<38:35>
                                           #RA/RB LOCATIONS
           R0=0
           R1=1
R2=2
R3=3
           R4=4
           R5≖5
           R6=6
R7=7
                                           #R12 = ARGUMENT LIST POINTER
           AP=0C
                                           #R13 = STACK FRAME POINTER
#R14 = STACK POINTER
           FF=OD
           SP=0E
                                           #R15 = PC, TO SOFTWARE, SCRATCH TO UCODE
           R15=0F
                                           #RC LOCATIONS
SPO.RC/=<38:35>
          T0=0
           T1=1
           T2=2
           T3=3
           T4=4
T5=5
           T6=6
           T7=7
           LC.SV=8
                                           ≯MEM MGMT SAVES LC HERE
           VA.SV=9
          PTE.VA=OA
PTE.PA=OB
          PC.SV=0C
           SC.SV=OD
VA.REF=OE
          MBIT.VA=OF
          PTE.MASK=OF
SUB/=<65:64>,.DEFAULT=0
                                           #SUBROUTINE CONTROL
                                           # DEFAULT
# PUSH UPC OF THIS MICROINSTRUCTION
# ONTO USTACK
# "OR" TOP OF USTACK TO UPC
# AND POP USTACK
          NOP=0
CALL=1
          RET=2
                                           #REPLACE LOW 8 BITS OF NEXT
# UPC WITH SPECIFIER DECODE FROM
# INSTRUCTION BUFFER
          SPEC=3
VAK/=<25:25>,.DEFAULT=0
                                         * #DEFAULT
          NOP=0
```

#LOAD VA

LOAD=1

.TOC *Machine definition : Validity checks*

.SET/VO=<.NOTC<NATIVE>J>
.SET/V1=<NATIVE>

.CREF FRE-ENABLE CROSS REFERENCE

```
. TOC
           *Macro definition
                                           : Resister transfer macros*
ALU_-1
                                            "AMX/RAMX.OXT,DT/LONG,ALU/NOTA"
ALU_O(A)
                                            "AMX/RAMX.OXT,DT/LONG,ALU/A"
ALU_0+D
                                            "AMX/RAMX.OXT.DT/LONG.RBMX/D.BMX/RBMX.ALU/A+B"
ALU_0+D+1
                                            "AMX/RAMX.OXT.DT/LONG.RBMX/D.BMX/RBMX.ALU/A+B+1"
                                            "KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/A+B"
"KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/A+B+1"
"AMX/RAMX.OXT,DT/LONG,BMX/LB,ALU/A+B+1"
ALU_O+KEJ+1
ALU_O+LB+1
ALU_O+LC+1
                                            "AMX/RAMX.OXT.DT/LONG.BMX/LC.ALU/A+B"
"AMX/RAMX.OXT.DT/LONG.BMX/LC.ALU/A+B+1"
                                            "AMX/RAMX.OXT.DT/LONG.BMX/MASK.ALU/A+B+1"
ALU_O+MASK+1
ALU_0+Q
ALU_0+Q+1
                                            "AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B"
"AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A+B+1"
                                            "AMX/RAMX.OXT.DT/LONG.RBMX/D.BMX/RBMX.ALU/A-B"
ALU_0-D
ALU_0-D-1
                                            "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B-1"
"AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B"
"KMX/@1,BMX/KMX,AMX/RAMX.OXT,DT/LONG,ALU/A-B-1"
ALULO-KEI
ALU_0-KEJ-1
                                             AMX/RAMX.OXT.DT/LONG.BMX/LB.ALU/A-B
ALU_O-LB
ALU_O-LC
                                            "AMX/RAMX.OXT.DT/LONG.BMX/LC.ALU/A-B"
ALU_O-LC-1
                                            "AMX/RAMX.OXT.DT/LONG.BMX/LC.ALU/A-B-1"
                                            "AMX/RAMX.OXT.DT/LONG.RBMX/Q.BMX/RBMX.ALU/A-B"
"AMX/RAMX.OXT.DT/LONG.RBMX/Q.BMX/RBMX.ALU/A-B-1"
ALU_0-0
ALU_0-0-1
                                            "ALU/@1;AMX/RAMX.OXT;LONG;BMX/RBMX;RBMX/D"
"ALU/@1;AMX/RAMX.OXT;LONG;BMX/CC"
ALU_OE JD
ALU_OCILC
                                            "RAMX/D, AMX/RAMX, ALU/A"
ALU_D
ALU_D(B)
                                            "RBMX/D,BMX/RBMX,ALU/B"
ALU_D+KE3
                                            "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B"
ALU_D+K[]+1
                                            "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1"
ALU..D+KEJ.RLOG
                                            "AMX/RAMX,RAMX/D,KMX/@1,BMX/KMX,ALU/A+B.RLOG"
ALU_D+LB
                                            "RAMX/D,AMX/RAMX,BMX/LB,ALU/A+B"
ALU_D+LC
                                            "RAMX/D, AMX/RAMX, BMX/LC, ALU/A+B"
ALU_D+LC+1
                                            "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+1"
                                            "RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1"
"ALU/A+B+PSL.C,AMX/RAMX,BMX/RBMX,RBMX/Q,RAMX/D"
ALU_D+LC+PSL.C
ALU_D+Q
ALU_D+Q+1
ALU_D+Q+PSL.C
ALU_D+RLOG
                                            "ALU/A+B,AMX/RAMX,RAMX/D,BMX/O,MSC/READ.RLOG"
ALU_D-KE]
                                            "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/A-B"
                                            "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1"
"RAMX/D,AMX/RAMX,BMX/LB,ALU/A-B"
ALU_D-KEJ-1
ALU_D-LB
                                            "RAMX/D,AMX/RAMX,BMX/LB,ALU/A-B.RLOG"
ALU_D-LB.RLOG
ALU_D-LC
ALU_D-LC-1
                                            "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B"
"RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B-1"
ALULD-Q
                                            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B"
                                            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B-1"
"RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A"
ALU_D-Q-1
ALU_D.OXTE3
ALU_D.OXTC3+KC3
                                            "RAMX/D, AMX/RAMX, OXT, DT/@1, KMX/@2, BMX/KMX, ALU/A+B"
ALU_D.OXTEJ+LC
                                            "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/LC
ALU..D.OXTE3+Q
                                            "ALU/A+B,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/RBMX,RBMX/Q"
ALU_D.OXTE3-KE3
                                            "RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B"
ALU_D.OXTE3-Q
ALU_D.OXTE3-AND.KE3
                                            "RAMX/D;AMX/RAMX.OXT;DT/@1;RBMX/Q;BMX/RBMX;ALU/A-B"
                                            "RAMX/D, AMX/RAMX.OXT, DT/@1, KMX/@2, BMX/KMX, ALU/AND"
ALU..D.OXTE3.ANDNOT.KE3
ALU..D.OXTE3.OR.Q
ALU..D.AND.KE3
                                            "ALU/ANDNOT,AMX/RAMX.OXT,DT/@1,RAMX/D,BMX/KMX,KMX,KMX/@2"
"RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,ALU/OR"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND"
ALU_D.AND.MASK
                                            "RAMX/D, AMX/RAMX, BMX/MASK, ALU/AND"
ALU...D.ANDNOT.KE3
                                            "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/ANDNOT"
ALULD. ANDNOT. MASK
                                            "RAMX/D, AMX/RAMX, BMX/MASK, ALU/ANDNOT
ALU_D.ANDNOT.Q
                                            "RAMX/D, AMX/RAMX, RBMX/Q, BMX/RBMX, ALU/ANDNOT"
ALU_D.OR.KE3
                                            "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/OR"
                                            "RAMX/D, AMX/RAMX, BMX/LC, ALU/OR"
ALUE DAORAL C
ALU_D.OR.Q
                                            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR"
```

```
"RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR"
ALU_D.OR.RCEJ
ALU_D.ORNOT.MASK
                                        "RAMX/D, AMX/RAMX, BMX/MASK, ALU/ORNOT
ALU_D.SXT[]
                                        "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A"
ALU_D.SXTE3+KE3
                                        "RAMX/D;AMX/RAMX.SXT;DT/@1;KMX/@2;BMX/KMX;ALU/A+B"
                                        "RAMX/D,AMX/RAMX.SXT,DT/@1,BMX/RBMX,ALU/A+B"
ALU_D.SXT[]+Q
                                        "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/ANDNOT,BMX/KMX,KMX/02"
"RAMX/D,AMX/RAMX.SXT,DT/@1,KMX/02,BMX/KMX,ALU/AND"
ALU_D.SXTEJ.ANDNOT.KEJ
ALU_D.SXTEJ.AND.KEJ
ALU_D.XOR.KEJ
                                        "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR"
"RAMX/D,AMX/RAMX,BMX/LC,ALU/XOR"
ALU_D.XOR.LC
ALU_D.XOR.Q
ALU_D.XOR.RCE3
                                        "RAMX/D, AMX/RAMX, RBMX/Q, BMX/RBMX, ALU/XOR"
                                        "RAMX/D, AMX/RAMX, SPO.R/LOAD.LC, SPO.RC/@1, BMX/LC, ALU/XOR"
ALU_D.XOR.REJ
                                        "RAMX/D, AMX/RAMX, SPO.R/LOAD.LAB, SPO.RAB/@1, BMX/LB, ALU/XOR"
ALU..DEJKEJ
                                        "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/@1"
ALU.DEJLB
                                        "ALU/@1,AMX/RAMX,RAMX/D,BMX/LB"
                                        "RAMX/D,AMX/RAMX,BMX/LC,ALU/@1"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1"
ALU_DEJLC
ALU_DEJQ
                                        *KMX/@1,BMX/KMX,ALU/B
ALU..LA
                                        "AMX/LA,ALU/A"
ALU_LA+KE3
                                        "AMX/LA,KMX/@1,BMX/KMX,ALU/A+B"
ALU_LA+K[]+1
                                        "ALU/A+B+1,AMX/LA,BMX/KMX,KMX/@1"
ALU_LA+KEJ.RLOG
                                        "AMX/LA,KMX/@1,BMX/KMX,ALU/A+B.RLOG"
ALU_LA+LB
                                        "AMX/LA,BMX/LB,ALU/A+B"
ALU_LA+LC
                                        "ALU/A+B, AMX/LA, BMX/LC
                                        "ALU/A+B+1,AMX/LA,BMX/LC"
ALU_LA+LC+1
ALU_LA+LC+PSL+C
                                        "ALU/A+B+FSL.C,AMX/LA,BMX/LC"
"ALU/A+B,AMX/LA,BMX/RBMX,RBMX/C"
ALU_LA+Q
ALU_LA-D
                                        *AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B
ALU_LA-D-1
ALU_LA-KEJ
                                        "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B-1"
                                        "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B
ALU_LA-KEJ-1
                                        "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B-1"
ALU_LA-KEJ.RLOG
                                        "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B.RLOG"
                                        "ALU/A-B,AMX/LA,BMX/LC"
"ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q"
"ALU/A-B-1,AMX/LA,BMX/RBMX,RBMX/Q"
"ALU/A-B-1,AMX/LA,BMX/RBMX,RBMX/Q"
ALU_LA-LC
ALULLA-Q
ALU_LA-Q-1
ALU_LA.AND.KEJ
ALU_LA.AND.LC
                                        "ALU/AND, AMX/LA, BMX/LC"
ALU_LA.ANDNOT.KEJ
                                        "AMX/LA,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_LA.ANDNOT.MASK
                                        "AMX/LA,BMX/MASK,ALU/ANDNOT"
ALU_LA.OR.KEJ
                                        "ALU/OR,AMX/LA,BMX/KMX,KMX/@1"
ALU_LA.XOR.LC
                                        "AMX/LA,BMX/LC,ALU/XOR"
ALU_LACID
                                        "AMX/LA,RBMX/D,BMX/RBMX,ALU/@1"
ALU_LACJLB
                                        "AMX/LA,BMX/LB,ALU/@1"
ALU_LAEJQ
                                        "AMX/LA,RBMX/Q,BMX/RBMX,ALU/@1"
"BMX/LB,ALU/R"
"BMX/LC,ALU/R"
ALULLB
ALU_LC
ALU_NOT.D
                                        "ALU/NOTA,AMX/RAMX,RAMX/D"
ALU_NOT.KEJ
                                        "BMX/KMX,KMX/@1,ALU/ORNOT,AMX/RAMX.OXT,DT/LONG"
ALU..NOT.RCEJ
                                        "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT"
ALU_PACK.FP
                                        "BMX/PACKED.FL,ALU/B"
ALU_PC
                                        *BMX/PC,ALU/R*
ALU_Q
                                        "RAMX/Q;AMX/RAMX;ALU/A"
                                        "RBMX/Q,BMX/RBMX,ALU/B"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B"
ALU..Q(B)
ALU_Q+K[]
                                        "ALU/A+B+1,ANX/RAMX,RAMX/Q,BMX/KMX,KMX/@1"
ALU_Q+KEJ+1
                                        "RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B"
"RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B+1"
ALU_Q+LB
ALU_Q+LB+1
                                        "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B"
ALU_Q+LC
                                        "ALU/A+B+1,ANX/RAMX,RAMX/Q,BMX/LC"
"ALU/A+B+PSL.C,AMX/RAMX,RAMX/Q,BMX/LC"
ALU_Q+LC+1
ALU_Q+LC+PSL.C
ALU_Q+MASK
                                        "ALU/A+B,AMX/RAMX,RAMX/Q,BMX/MASK"
ALU_Q-D
                                        "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B"
                                        "ALU/A-B-1,ANX/RAMX,RAMX/Q,BMX/RBMX,RBMX/D"
ALU_Q-D-1
ALU_Q-K[]
                                        "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B"
```

```
"RAMX/Q,AMX/RAMX,BMX/LB,ALU/A-B
ALU_Q-LB
ALU_Q-LC
                                                 "RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B"
                                                 "ALU/A-B-1, AMX/RAMX, RAMX/Q, BMX/MASK"
ALU_Q-MASK-1
ALU_Q.OXTE3
                                                 "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A"
                                                 "RAMX/Q;AMX/RAMX.OXT;DT/@1;ALU/A"
"ALU/A+B;AMX/RAMX.OXT;DT/@1;BMX/RBMX;RBMX/D;RAMX/Q"
"ALU/A+B;AMX/RAMX.OXT;DT/@1;BMX/RBMX;RAMX/Q;RBMX/D"
"ALU/A+B;AMX/RAMX.OXT;DT/@1;RAMX/Q;BMX/KMX;KMX/@2"
"ALU/A-B;RAMX/Q;AMX/RAMX.OXT;DT/@1;BMX/RBMX"
ALU_Q.OXTE3+D
ALU_Q.OXTEJ+D+1
ALU_Q.OXTCJ-KCJ
ALU_Q.OXTCJ-D
ALU_Q.OXTCJ-KCJ
ALU_Q.OXTCJ-KCJ
ALU_Q.OXTCJ-GNDNOT-KCJ
                                                  "ALU/A-B,AMX/RAMX.OXT,DT/@1,RAMX/Q,BMX/KMX,KMX/@2"
                                                 "ALU/ANDNOT,AMX/RAMX.OXT,DT/@1;RAMX/G,BMX/KMX;KMX/@2"
"ALU/OR,AMX/RAMX.OXT,DT/@1;RAMX/G,BMX/KMX;KMX/@2"
"ALU/OR,AMX/RAMX.OXT,DT/@1;RAMX/G,BMX/RBMX,RBMX/D"
"ALU/OR,AMX/RAMX.OXT,DT/@1;RAMX/G,BMX/RBMX,RBMX/D"
"AMX/RAMX,RAMX/G,BMX/RBMX;RBMX/D,ALU/AND"
ALU_Q.OXT[].OR.D
ALU_Q.AND.D
ALU_Q.AND.KEJ
                                                 "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND"
ALU_Q.ANDNOT.KCJ
                                                 "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_Q.ANDNOT.MASK
ALU_Q.ANDNOT.REJ
                                                  RAMX/Q, AMX/RAMX, BMX/MASK, ALU/ANDNOT
                                                 "ALU/ANDNOT, AMX/RAMX, RAMX/Q, BMX/LB, SPO.R/LOAD.LAB, SPO.RAB/@1"
                                                 "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR"
ALU_Q.OR.KEJ
                                                 "RAMX/Q;AMX/RAMX;BMX/LC;ALU/OR"
"ALU/ORNOT;AMX/RAMX;RAMX/Q;BMX/KMX;KMX/@1"
ALU_Q.OR.LC
ALU_Q.ORNOT.KEJ
ALU_Q.SXT[]
                                                 "ALU/A,AMX/RAMX.SXT,DT/@1,RAMX/Q"
ALU_Q.SXT[]+K[]
ALU_Q.SXT[]+LB
ALU_Q.SXT[]+LB+1
ALU_Q.SXT[]+PC
ALU_Q.SXT[].ANDNOT.K[]
                                                 "RAMX/Q;AMX/RAMX.SXT,DT/@1,KMX/@2,BMX/KMX;ALU/A+B"
"RAMX/Q;AMX/RAMX.SXT,DT/@1,BMX/LB;ALU/A+B"
                                                 "RAMX/Q, AMX/RAMX.SXT, DT/@1, BMX/LB, ALU/A+B+1"
                                                 "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/FC,ALU/A+B
                                                 "ALU/ANDNOT, AMX/RAMX.SXT, RAMX/Q, BMX/KMX, KMX/@2,DT/@1"
ALU_Q.XOR.D
                                                 "RAMX/Q,AMX/RAMX,BMX/RBMX,RBMX/D,ALU/XOR"
ALU_Q.XOR.KEJ
ALU_Q.XOR.LC
                                                 "RAMX/G,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XDR"
"RAMX/G,AMX/RAMX,BMX/LC,ALU/XDR"
                                                 "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/XOR"
"RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/@1"
ALU_Q.XOR.RCC3
ALU_QCJD
                                                 "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST,AMX/LA,ALU/A"
ALU_R(DST)
                                                 "SPO.AC/LOAD.LAB,SPO.ACN/SC,AMX/LA,KMX/@1,BMX/KMX,ALU/ANDNOT"
ALU_R(SC).ANDNOT.KE3
                                                 "SPO.AC/LOAD.LAB,SPO.ACN/SF1.SP1,AMX/LA,KMX/@1,BMX/KMX,ALU/A+B.RLOG"
"SPO/LOAD.LC.SC,BMX/LC,ALU/B"
ALU_R(SP1)+KEJ.RLOG
ALU_RC(SC)
                                                 "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B"
ALU_RCEJ
ALU_RLOG
                                                 "BMX/O, ALU/B, MSC/READ. RLOG"
ALU.REJ
                                                 "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A"
                                                 "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/A-B"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/A-B"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT"
ALU_RCJ-KCJ
ALU_REJ.AND.KEJ
ALU_REJ.AND.LC
ALU_REJ.ANDNOT.KEJ
ALU_REJ.ANDNOT.MASK
ALU_REJ.OR.KEJ
                                                 "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/OR"
ALULREJ.ORNOT.KEJ
                                                 "ALU/ORNOT,AMX/LA,BMX/KMX,SPO.R/LOAD.LAB,SPO.RAB/@1,KMX/@2"
                                                 "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/XOR"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,RBMX/Q,BMX/RBMX,ALU/XOR"
ALU_REJ.XOR.KEJ
ALU_REJ.XOR.Q
CACHE . P. DE 3
                                                 "VAK/NOP, MCT/WRITE.P, DT/@1, DK/NOP"
                                                 "VAK/NOP, MCT/WRITE.V. WCHK, MSC/@1, DK/NOP"
CACHECILD
                                                 "MCT/EXTWRITE.P,LONG,VAK/NOP,DK/NOP"
CACHE_D(QUAD)
                                                 "VAK/NOP, MCT/WRITE.V.WCHK, DT/INST.DEP, DK/NOP"
"VAK/NOP, MCT/WRITE.V.WCHK, DT/@1, DK/NOP"
CACHE_D.INST.DEP
CACHE_DED
                                                 "VAK/NOP, MCT/LOCKWRITE.V.XCHK,DT/@1,DK/NOP"
"VAK/NOP, MCT/WRITE.V.NOCHK,DT/@1,DK/NOP"
CACHELDED.LK
CACHELDE3.NOCHK
                                                 "RAMX/D, AMX/RAMX, RBMX/Q, BMX/RBMX, ALU/A+B, SHF/ALU, DK/SHF, QK/SHF"
D&Q_D+Q
                                                 *BMX/PC,ALU/B,SHF/ALU,DK/SHF,SPO.R/WRITE.RC,SPO.RC/01*
*VAK/LOAD,SHF/ALU,DK/SHF*
D&RCCJ_PC
D&VA_ALU
                                                 "RAMX/D, AMX/RAMX, BMX/LC, ALU/A+B, VAK/LOAD, SHF/ALU, DK/SHF"
D&VA_D+LC
                                                 "D_D+Q,VAK/LOAD"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,DK/SHF"
D&VA_D+Q
D&VA_D-KEJ
D&VA...LA
                                                 "AMX/LA, ALU/A, VAK/LOAD, SHF/ALU, DK/SHF"
```

```
D&VA_LB
                                            "BMX/LB,ALU/B,VAK/LOAD,SHF/ALU,DK/SHF"
"RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,DK/Q"
DAVA_Q
D&VA_Q+LB.PC
                                            "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,DK/SHF"
                                            "VAK/NOP, MCT/READ. V. RCHK, DT/@1, DK/NOP"
DE3...CACHE
DEJ_CACHE.IBCHK
                                            "VAK/NOP, MCT/READ. V. IBCHK, DT/@1, DK/NOP"
DCJ_CACHE.LK
                                            "VAK/NOP, MCT/LOCKREAD. V. WCHK, DT/@1, DK/NOP"
                                            "VAK/NOP, MCT/READ.V.NOCHK, DT/@1, DK/NOP"
"VAK/NOP, MCT/READ.P, DT/@1, DK/NOP"
"VAK/NOP, MCT/READ.V.WCHK, DT/@1, DK/NOP"
DEJ_CACHE.NOCHK
DIJ_CACHE.P
DEJ_CACHE.WCHK
D_0
D_0+KE3+1
                                            "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_O+LC+1
                                            "AMX/RAMX.OXT.DT/LONG.BMX/LC.ALU/A+B+1.SHF/ALU.DK/SHF
                                            "AMX/RAMX.OXT.DT/LONG.RBMX/D.BMX/RBMX.ALU/A-B.SHF/ALU.DK/SHF"
D_0-D
D_O-KEJ
                                            "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
D_0-Q
                                            "AMX/RAMX.OXT.DT/LONG.RBMX/Q.BMX/RBMX.ALU/A-B.SHF/ALU.DK/SHF"
D_0-Q-1
                                            "ALU_O-Q-1,D_ALU"
"DK/ACCEL,ACF/SYNC"
"SHF/ALU,DK/SHF"
D_ACCEL &SYNC
D_ALU
D_ALU(FRAC)
                                            "SHF/ALU,DK/SHF.FL"
                                            "SHF/LEFT, DK/SHF"
D_ALU.LEFT
D_ALU.LEFT2
                                            "SHF/ALU.DT,DT/LONG,DK/SHF"
D_ALU.LEFT3
                                            "SHF/LEFT3,DK/SHF"
D_ALU.RIGHT
                                            SHF/RIGHT, DK/SHF
                                            "SHF/RIGHT2,DK/SHF"
D_ALU.RIGHT2
D_BLANK
                                            *D_K[.20]*
                                            "VAK/NOP,MCT/READ.V.IBCHK,DT/INST.DEP,DK/NOP"
"VAK/NOP,MCT/LOCKREAD.V.WCHK,MSC/@1,DK/NOP"
"VAK/NOP,MCT/READ.V.WCHK,MSC/@1,DK/NOP"
"VAK/NOP,MCT/READ.V.RCHK,MSC/@1,DK/NOP"
D_CACHE.INST.DEP
D_CACHE.LKEJ
D_CACHE.WCHK[]
D_CACHE[]
D_D(FRAC)
                                            "RAMX/D, AMX/RAMX, ALU/A, SHF/ALU, DK/SHF, FL"
D_D+KE3
                                            "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/A+B, SHF/ALU, DK/SHF"
D_D+KE3+1
                                            "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,DK/SHF"
                                            "RAMX/D,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF"
"RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,DK/SHF"
"RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B+PSL.C,SHF/ALU,DK/SHF"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF"
D_-D+LB
n net c
D_D+LC+PSL.C
D_D+Q
D_D+Q+1
                                            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B+1,SHF/ALU,DK/SHF"
D_D-KC3
                                            "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/A-B, SHF/ALU, DK/SHF"
D_D-LC
                                            "RAMX/D, AMX/RAMX, BMX/LC, ALU/A-B, SHF/ALU, DK/SHF"
D_D-Q
                                            "RAMX/D, AMX/RAMX, RBMX/Q, BMX/RBMX, ALU/A-B, SHF/ALU, DK/SHF"
D_D-0-1
                                            "RAMX/D, AMX/RAMX, RBMX/Q, BMX/RBMX, ALU/A-B-1, SHF/ALU, DK/SHF"
D_D.OXTE3
                                            "RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_D.OXTCJ+KCJ
                                            "RAMX/D,AMX/RAMX.0XT,DT/@1,KMX/@2,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
                                            "RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RBMX/Q,D_ALU"
"RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,RBMXPD,ALU"
"RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
"RAMX/D,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.OXTE3+Q
D_D.OXTE3+Q+1
D_D.OXTE3.ANDNOT.KE3
D_D.OXTE3.OR.Q
D_D.OXTCJ.XOR.Q
                                            "DK/SHF,ALU/XOR,SHF/ALU,AMX/RAMX.OXT,RAMX/D,DT/@1,RBMX/Q,BMX/RBMX"
D_D.OXTE3.XOR.RCE3
                                            *RAMX/D,AMX/RAMX.OXT,DT/@1,SPO.R/LOAD.LC,SPO.RC/@2,BMX/LC,ALU/XOR,SHF/ALU,DK/SHF
D_D.AND.KC]
                                            "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
D_D.AND.KCJ.LEFT2
                                            "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU.DT,DT/LONG,DK/SHF"
D_D.AND.KCJ.RIGHT
                                            "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,DK/SHF"
D_D.AND.LC
D_D.AND.MASK
                                            RAMX/D, AMX/RAMX, BMX/LC, ALU/AND, SHF/ALU, DK/SHF **RAMX/D, AMX/RAMX, BMX/MASK, ALU/AND, SHF/ALU, DK/SHF
D_D.AND.Q
                                            "RAMX/D, AMX/RAMX, RBMX/Q, BMX/RBMX, ALU/AND, SHF/ALU, DK/SHF"
                                            "RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.AND.RCC3
D_D.ANDNOT.KC3
D_D.ANDNOT.LC
                                            "RAMX/D, AMX/RAMX, BMX/LC, ALU/ANDNOT, SHF/ALU, DK/SHF"
D_D.ANDNOT.PSWZ
                                            "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.4,SHF/ALU"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_D.ANDNOT.Q
D_D.ANDNOT.RCEJ
                                            *RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,DK/SHF*
```

```
"DK/LEFT"
D_D.LEFT
D_D.LEFT2
                                               "DK/LEFT2"
                                               "D_D.OR.KE.303"
D_D.OR.ASCII
                                               "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/OR, SHF/ALU, DK/SHF"
D_D.OR.KEJ
                                               "DK/SHF, ALU/OR, AMX/RAMX, RAMX/D, BMX/KMX, KMX/.1, SHF/ALU"
D_D.OR.PSWC
D_D.OR.PSWV
                                               "DK/SHF,ALU/OR,AMX/RAMX,RAMX/D,BMX/KMX,KMX/.2,SHF/ALU"
                                                "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,DK/SHF"
D_D.OR.Q
                                               "RAMX/D, AMX/RAMX, BMX/MASK, ALU/ORNOT, SHF/ALU, DK/SHF"
"RAMX/D, AMX/RAMX, BMX/MASK, ALU/ORNOT, SHF/ALU, DK/SHF"
"RAMX/D, AMX/RAMX, BMX/MASK, ALU/ORNOT, SHF/ALU, DK/SHF"
"DK/RIGHT"
D_D.OR.RCE3
D_D.OR.REJ
D_D.ORNOT.MASK
D_D.RIGHT
D_D.RIGHT(B)
                                                "RBMX/D,BMX/RBMX,ALU/B,SHF/RIGHT,DK/SHF"
D_D.RIGHT2
                                               *DK/RIGHT2
                                                "DK/BYTE.SWAP"
D_D.SWAP
                                                "RAMX/D, AMX/RAMX.SXT, DT/@1, ALU/A, SHF/ALU, DK/SHF"
D_D.SXTE3
                                               "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/RIGHT,DK/SHF"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/XOR,SHF/ALU,DK/SHF"
D_D.SXTEJ.RIGHT
D_D.XOR.KEJ
                                                "RAMX/D, AMX/RAMX, BMX/LC, ALU/XOR, SHF/ALU, DK/SHF"
D_D.XOR.LC
                                                "RAMX/D, AMX/RAMX, RBMX/Q, BMX/RBMX, ALU/XOR, SHF/ALU, DK/SHF"
D_D.XOR.Q
D_DAL.NORM
                                                "DK/DAL.SV"
D_DAL .SC
                                                "DK/DAL.SC"
                                               "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/@1,SHF/ALU,DK/SHF"
"RAMX/D,AMX/RAMX,BMX/MASK,ALU/@1,SHF/ALU,DK/SHF"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1,SHF/ALU,DK/SHF"
DEDECKES
D_DE JMASK
D_DCJQ
                                                "MCT/READ.INT.SUM,DK/NOP"
"KMX/@1,BMX/KMX,ALU/B,SHF/ALU,DK/SHF"
"KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,DK/SHF"
D_INT.SUM
DLKCD
D_KCJ.RIGHT
                                               "KMX/01, BMX/KMX, ALU/B, SHF/RIGHT2, DK/SHF"
"AMX/LA, ALU/A, SHF/ALU, DK/SHF"
D_KCJ.RIGHT2
D_LA
                                                "AMX/LA,ALU/A,SHF/ALU,DK/SHF.FL"
I)_LA(FRAC)
                                               "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B+PSL.C,SHF/ALU,DK/SHF"
"DK/SHF,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/D,SHF/ALU"
D_LA+D+PSL.C
D_LA-D
                                               "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF
D_LA-KED
                                               "AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
"AMX/LA,ALU/A,SHF/RIGHT,DK/SHF"
"BMX/LB,ALU/B,SHF/ALU,DK/SHF"
D_LA.AND.KE3
D_LA.RIGHT
DLLB
                                               "BMX/PC.OR.LB,ALU/B,SHF/ALU,DK/SHF"
"BMX/LC,ALU/B,SHF/ALU,DK/SHF"
D.LB.PC
D_LC
D_LC(FRAC)
                                                "BMX/LC,ALU/B,SHF/ALU,DK/SHF.FL"
                                                "RAMX/D, AMX/RAMX, ALU/NOTA, SHF/ALU, DK/SHF"
D_NOT.D
                                               "RAMX/D; AMX/RAMX; ALU/NOTA; SHF/ALU; DK/SHF"
"KMX/@1; BMX/KMX; AMX/RAMX; OXT; DT/LONG; ALU/ORNOT; SHF/ALU; DK/SHF"
"BMX/MASK; AMX/RAMX; OXT; DT/LONG; ALU/ORNOT; SHF/ALU; DK/SHF"
D_NOT.KE3
D_NOT.MASK
                                               "RAMX/Q,AMX/RAMX,ALU/NOTA,SHF/ALU,DK/SHF'
"LA_RAC@1],AMX/LA,ALU/NOTA,D_ALU"
"BMX/PACKED.FL,ALU/B,SHF/ALU,DK/SHF"
D_NOT+Q
D_NOT.RE3
D_PACK . FP
                                               *BMX/PACKED.FL,ALU/B,SHF/LEFT,DK/SHF*
*BMX/PC,ALU/B,SHF/ALU,DK/SHF*
D_PACK.FP.LEFT
D_PC
                                               "BMX/PC,ALU/B,SHF/LEFT,DK/SHF"
D_PC.LEFT
                                               "DK/Q"
D_{-}Q
                                               "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,DK/SHF.FL"
D_Q(FRAC)
                                               "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,DK/SHF"
D_Q+D
                                               "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,DK/SHF"
"RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,SHF/ALU,DK/SHF"
D_Q+KE3
D_Q+LB
                                               "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,DK/SHF"
D..Q+PC
                                               "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,DK/SHF"
"RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B-1,SHF/ALU,DK/SHF"
D_Q-D
D_Q-D-1
D_Q-KEJ
                                                "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,DK/SHF"
                                                "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,DK/SHF"
n n-KET-1
                                                "RAMX/Q,AMX/RAMX,BMX/O,MSC/READ.RLOG,ALU/A-B,SHF/ALU,DK/SHF"
D_Q-PCSV
                                               "RAMX/Q,AMX/RAMX,OXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF"
"RAMX/Q,AMX/RAMX,BMX/LC,ALU/AND,SHF/ALU,DK/SHF"
D_Q.OXTE3
D_Q.AND.KEJ
D.O.AND.LC
                                                "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,DK/SHF"
D.Q.AND.MASK
```

```
D_Q.AND.RCEJ
                                              *RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,DK/SHF*
D_Q.ANDNOT.D
                                              "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,DK/SHF
D_Q.ANDNOT.KCJ
                                              "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,SHF/ALU,DK/SHF"
D_Q.ANDNOT.MASK
                                              "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ANDNOT,SHF/ALU,DK/SHF"
                                              "DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/1,SHF/ALU"
"DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU"
"DK/SHF,ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.4,SHF/ALU"
"RAMX/Q,AMX/RAMX,ALU/A,SHF/LEFT,DK/SHF"
"RAMX/Q,AMX/RAMX,KMX/Q1,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF"
D_Q.ANDNOT.PSWC
D_Q.ANDNOT.PSWN
D_Q.ANDNOT.PSWZ
D_Q.LEFT
D_Q.OR.KEJ
D_Q.OR.PSWC
D_Q.OR.RCEJ
                                              *DK/SHF,ALU/OR,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/.1,SHF/ALU*
*RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/Q1,BMX/LC,ALU/OR,SHF/ALU,DK/SHF*
                                              "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/ORNOT,SHF/ALU,DK/SHF"
"RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT,DK/SHF"
"RAMX/Q,AMX/RAMX,ALU/A,SHF/RIGHT2,DK/SHF"
D_Q.ORNOT.MASK,
D_Q.RIGHT
D_Q.RIGHT2
                                              "RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,DK/SHF"
D_Q.SXT[]
                                              "RAMX/Q;AMX/RAMX;SPO.R/LOAD.LC;SPO.RC/@1;BMX/LC;ALU/XOR;SHF/ALU;DK/SHF"
"RAMX/Q;AMX/RAMX;RBMX/D;BMX/RBMX;ALU/@1;SHF/ALU;DK/SHF;
"ALU/@1;SHF/ALU;DK/SHF;BMX/KMX;KMX/@2;AMX/RAMX;RAMX/Q"
"RAMX/Q;AMX/RAMX;BMX/MASK;ALU/@1;SHF/ALU;DK/SHF"
D_G.XOR.RCC3
D_GC3D
DEGECKEC
D_QC JMASK
D_R(PRN+1)
                                              "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_R(SC)
                                              "SPO.AC/LOAD.LAB.SPO.ACN/SC.AMX/LA.ALU/A.SHF/ALU.DK/SHF"
                                              "SPO.AC/LOAD.LAB,SPO.ACN/SP1+1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
"SPO.AC/LOAD.LC.SC,BMX/LC,ALU/B,SHF/ALU,DK/SHF"
"SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SHF/ALU,DK/SHF"
"BMX/O,MSC/READ.RLOG,ALU/B,SHF/ALU,DK/SHF"
"BMX/O,MSC/READ.RLOG,ALU/B,SHF/RIGHT,DK/SHF"
D_R(SP1+1)
D_RC(SC)
D_RLOG
D_RLOG.RIGHT
DLRED
                                              "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF"
D_REJ(FRAC)
                                              "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,DK/SHF.FL"
D_RCJ.AND.KCJ
                                              *SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND,SHF/ALU,DK/SHF*
                                              *SPO.R/LOAD.LAB,SPO.RAB/01,AMX/LA,KMX/02,BMX/KMX,ALU/OR,SHF/ALU,DK/SHF*
D_RCJ.OR.KCJ
D_RCJ.ORNOT.KCJ
                                              "LAB_RC@13,AMX/LA,BMX/KMX,KMX/@2,ALU/ORNOT,D_ALU"
EALU_D(EXP)
                                              "RAMX/D, AMX/RAMX, EBMX/AMX, EXP, EALU/B"
EALULFE
EALULKED
                                              "EBMX/FE,EALU/B"
"KMX/G1,EBMX/KMX,EALU/B"
EALU_RCJ(EXP)
                                              "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,EBMX/AMX.EXP,EALU/B"
EALU_SC
                                              "EALU/A"
EALULSC+FE
                                              "EBMX/FE, EALU/A+B"
EALU_SC+KC3
                                              "KMX/@1,EBMX/KMX,EALU/A+B"
                                              "EBMX/FE,EALU/A-B"
"KMX/01,EBMX/KMX,EALU/A-B"
"KMX/01,EBMX/KMX,EALU/ANDNOT"
EALULSC-FE
EALU_SC-KED
EALU_SC.ANDNOT.KEJ
EALU_STATE
                                              "EALU/A, MSC/LOAD. STATE"
FE&SCLKED
                                              "KMX/@1,EBMX/KMX,EALU/B,FFK/LOAT,SMX/FALU,SCK/LOAT."
FE_O(A)
                                              "AMX/RAMX.OXT.DT/LONG.EBMX/AMX.EXP.EALU/B.FEK/LOAD"
FELD(EXP)
                                              "RAMX/D, AMX/RAMX, EBMX/AMX, EXF, EALU/B, FEK/LOAD"
FELEALU
                                              "FEK/LOAD"
FE_KCJ
                                              *KMX/@1,EBMX/KMX,EALU/B,FEK/LOAD*
FE_LA(EXP)
                                              "AMX/LA,ERMX/AMX.EXP,EALU/B,FEK/LOAD"
FE_NABS(SC-FE)
                                              "EALU/NABS.A-B,EBMX/FE,FEK/LOAD"
                                              "AMX/LA,EBMX/AMX.EXP,EALU/NABS.A-B,FEK/LOAD"
"RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,FEK/LOAD"
FE_NABS(SC-LA(EXP))
FE_Q(EXP)
                                              "SPO.R/LOAD.LAE,SPO.RAB/01,AMX/LA,EBMX/AMX.EXP,EALU/B,FEK/LOAD"
"EALU/A,FEK/LOAD"
FELRED(EXP)
FE_SC
FE_SC+1
                                              "EALU/A+1, FEK/LOAD"
FE_SC+FE
                                              "EBMX/FE, EALU/A+B, FEK/LOAD"
                                              "KMX/@1,EBMX/KMX,EALU/A+B,FEK/LOAD"
"AMX/LA,EBMX/AMX.EXP,EALU/A+B,FEK/LOAD"
FE.SC+KEJ
FE_SC+LA(EXP)
FE_SC-FE
                                              "EBMX/FE,EALU/A-B,FEK/LOAD"
FE_SC-KEJ
FE_SC-LA(EXP)
                                              "KMX/@1,EBMX/KMX,EALU/A-B,FEK/LOAD"
                                              "AMX/LA,EBMX/AMX.EXP,EALU/A-B,FEK/LOAD"
FE_SC-SHF.VAL
                                              "EBMX/SHF.VAL, EALU/A-B, FEK/LOAD"
```

```
"EBMX/FE,EALU/ANDNOT,FEK/LOAD"
"KMX/01,EBMX/KMX,EALU/ANDNOT,FEK/LOAD"
"EALU/OR,EEMX/KMX,KMX,01,FEK/LOAD"
"EBMX/SHF.VAL,EALU/B,FEK/LOAD"
 FE_SC.ANDNOT.FE
FE_SC.ANDNOT.KEJ
FE_SC.OR.KEJ
 FE_SHF . VAL
 FE_STATE
                                                                                  "MSC/LOAD.STATE, EALU/A, FEK/LOAD"
                                                                                   "CID/WRITE.SC"
  ID(SC)_D
                                                                                   "CID/WRITE.KMX,ID.ADDR/@1"
 IDCILD
  ID_D&NO.SYNC
                                                                                  "CID/WRITE.KMX,ADS/IBA,KMX/SP1.CON"
 ID_D.SYNC
                                                                                  "CID/WRITE.KMX, ADS/IBA, KMX/SP1.CON, ACF/SYNC"
 KET
                                                                                  "KMX/@1"
 LAB ... R(DST)
                                                                                  "SPO.AC/LOAD.LAB,SPO.ACN11/DST.DST"
                                                                                  'SPO.AC/LOAD.LAB.SPO.ACN/PRN'
 LAB_R (PRN)
                                                                                  "SPO.AC/LOAD.LAB,SPO.ACN/PRN+1"
"SPO.AC/LOAD.LAB,SPO.ACN/SC"
 LAB_R(PRN+1)
 LAB_R(SC)
LAB_R(SP1)
                                                                                  "SPO.AC/LOAD.LAB.SPO.ACN/SP1.SP1"
 LAB_R(SP1+1)
LAB_R1&RCCJ_0
                                                                                  "SPO.AC/LOAD.LAB,SPO.ACN/SP1+1"
"ALU_O(A),LAB_R1&RCE@1J_ALU"
                                                                                 "ALU_A(A);LAB_R1&RCC@1]_ALU"

"ALU/A+B+1;AMX/RAMX.OXT;DT/LONG;BMX/LC;8PO.R/LOAD.LAB1.WRITE.RC;SPO.RC/@1;SHF/ALU"

"SPO.R/LOAD.LAB1.WRITE.RC;SPO.RC/@1;ALU/A-B;MX/RAMX.OXT;DT/LONG;BMX/RBMX;RBMX/D;SHF/ALU"

"SPO.R/LOAD.LAB1.WRITE.RC;SPO.RC/@1;SHF/RIGHT2"

"SPO.R/LOAD.LAB1.WRITE.RC;SPO.RC/@1;SHF/RIGHT2"

"ALU_D+LC;LAB_R1&RCC@1]_ALU"

"ALU_D+C;LAB_R1&RCC@1]_ALU"

"ALU_G-KC@2];LAB_R1&RCC@1]_ALU"
 LAB_R1%RCEJ_O+LC+1
 LAB_R1&RCEJ_O-D
LAB_R1&RCEJ_ALU
 LAB.R18RCCJ_ALU.RIGHT2
LAB.R18RCCJ_D+LC
LAB.R18RCCJ_D.OXTCJ+KCJ
 LAB_R18RCEJ_Q-KEJ
                                                                                  "SPO.R/LOAD.LAB,SPO.RAB/@1"
 LAB_RET
 LA_R(DST)&LB_R(SRC)
                                                                                 "SPO.AC/LOAD.LAB,SPO.ACN11/DST.SRC"
 LA_R(SP2)&LB_R(SP1)
                                                                                  "SPO.AC/LOAD.LAB,SPO.ACN/SP2.SP1"
"SPO.AC/LOAD.LA,SPO.RAB/@1"
 LA_RACI
                                                                                "SPO.AC/LOAD.LA,SPO.RAB/@1"
"SPO/LOAD.LC.SC"
"SPO.R/LOAD.LC.SC"
"AMX/LA,BMX/LB,ALU/A+B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
"AMX/LA,BMX/LB,ALU/A+B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
"AMX/LA,BMX/LB,ALU/A+B.RLOG,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
"AMX/LA,BMX/LB,ALU/A-B,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
"AMX/LA,BMX/LB,ALU/A-B.RLOG,SHF/LEFT,SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1"
"SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU"
"SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU"
"SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU,ALU/A+B,AMX/LA,BMX/KMX,KMX/@2"
"ALU_LA-KC@2],LC.RCC@1]&R1_ALU"
"ALU_LB,LC.RCC@1]&R1_ALU"
"SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU,ALU/A+B,AMX/RAMX,RAMX/RAMX/RAMX/@"
 LC_RC(SC)
LC_RCCJ
LC_RCCJ&R1_(LA+LB).LEFT
LC_RCC18R1_(LA+LB+PSL.C).LEFT
LC_RCC18R1_(LA+LB.RLOG).LEFT
LC_RCC18R1_(LA-LB).LEFT
LC_RCC18R1_(LA-LB.RLOG).LEFT
LC_RCC18R1_ALD
LC_RCC18R1_D
LC_RCEJ&RI_LA+KEJ
LC_RCEJ&RI_LA-KEJ
LC_RCEJ&RI_LB
LC_RCC3&R1_Q
                                                                                 "SPO.R/LOAD.LC.WRITE.RAB1,SPO.RC/@1,SHF/ALU,ALU/A,AMX/RAMX,RAMX/Q"
                                                                                 "CCK/NZ_ALU.VC_VC"
"CCK/NZ_ALU.VC_O"
"CCK/N_AMX.Z_TST.VC_VC"
N&Z_ALU.V&C_0
N_AMX.Z_TST
PC&VA...ALU
                                                                                  "VAK/LOAD,PCK/PC_VA
                                                                                "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX.OXT,DT/@1,ALU/A,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
"RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
"RAMX/@1,BMX/KMX,ALU/B,VAK/LOAD,PCK/PC_VA"
PC&VALD+KC3
PC8VALD-KET
PC8VA_D-PC
PC8VA_D.OXTC3
PC&VA_D.OXTE3+PC
PC&VA_D.SXTE3+PC
PC&VA_KE3
                                                                                 "BMX/PC,ALU/B,VAK/LOAD,PCK/PC_VA"
"RAMX/Q,AMX/RAMX,ALU/A,VAK/LOAD,PCK/PC_VA"
"RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
PC&VALPO
PC&VALQ
PC&VA_Q+PC
PC&VA_Q-D
                                                                                 "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
```

```
PC&VA_Q-KEJ
                                       "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,VAK/LOAD,PCK/PC_VA"
PC&VA_Q.SXTCJ+PC
                                       "RAMX/Q,AMX/RAMX.SXT,DT/@1,BMX/PC,ALU/A+B,VAK/LOAD,PCK/PC_VA"
                                       "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,VAK/LOAD,PCK/PC_VA"
PC&VA_RCE3
                                       *SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT,VAK/LOAD,PCK/PC__VA*
PC&VA..RCJ.ANDNOT.KCJ
PC_PC+1
                                       *PCK/PC+1
PC_PC+2
                                       "PCK/PC+2"
                                       *PCK/PC+4*
PC_PC+4
PC_PC+N
                                       *PCK/PC+N*
PC_Q+PC
                                       "ALU/A+B,VAK/LOAD,PCK/PC_VA,BMX/PC,AMX/RAMX,RAMX/Q"
PC_VA
PC_VIBA
                                       "PCK/PC_VA"
                                      "PCK/PC_IBA"
"CCK/C_AMXO"
PSL<C>_AMXO
Q8VA_ALU
                                       "VAK/LOAD, SHF/ALU, QK/SHF"
                                      "RAMX/D, AMX/RAMX, ALU/A, VAK/LOAD, SHF/ALU, QK/SHF"
Q&VA_D
Q&VA_D+LC
                                       "RAMX/D, AMX/RAMX, BMX/LC, ALU/A+B, VAK/LOAD, SHF/ALU, QK/SHF"
                                       "AMX/LA,ALU/A,VAK/LOAD,SHF/ALU,QK/SHF"
Q&VA_LA
Q&VA_Q+LB.PC
                                       "RAMX/Q,AMX/RAMX,BMX/PC.OR.LB,ALU/A+B,VAK/LOAD,SHF/ALU,QK/SHF"
QD_(Q+LB)D.RIGHT2
                                       "ALU_Q+LB,Q_ALU.RIGHT2,D_D.RIGHT2"
QD_(Q+LC)D.RIGHT2
QD_(Q-LB)D.RIGHT2
                                      "ALU_Q+LC,Q_ALU.RIGHT2,D_D.RIGHT2"
"ALU_Q-LB,Q_ALU.RIGHT2,D_D.RIGHT2"
QD_(Q-LC)D.RIGHT2
                                       "ALU_Q-LC,Q_ALU.RIGHT2,D_D.RIGHT2"
QD_QD.RIGHT2
                                       "ALU_Q,Q_ALU.RIGHT2,D_D.RIGHT2
Q_(LA+Q).RIGHT
                                       "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/RIGHT,QK/SHF"
Q_(Q+LB).RIGHT
                                       "RAMX/Q, AMX/RAMX, BMX/LB, ALU/A+B, SHF/RIGHT, QK/SHF"
                                       *QK/CLR
Q_{-}0
Q_0+LC+1
                                       "ALU/A+B+1,AMX/RAMX.OXT,DT/LONG,SHF/ALU,QK/SHF,BMX/LC"
Q_O+MASK+1
Q_O+PC.RLOG
                                       "AMX/RAMX.OXT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,QK/SHF"
"AMX/RAMX.OXT,DT/LONG,BMX/PC,ALU/A+B.RLOG,SHF/ALU,QK/SHF"
Q_0-D
                                       "AMX/RAMX.OXT.DT/LONG.RBMX/D.BMX/RBMX.ALU/A-B.SHF/ALU.QK/SHF"
                                       "AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
"AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
Q_O-KEJ
Q_O-LC
0.0-0
                                       "AMX/RAMX.OXT.DT/LONG.RBMX/Q.BMX/RBMX.ALU/A-B.SHF/ALU.QK/SHF"
Q_ACCEL &SYNC
                                       "QK/ACCEL, ACF/SYNC"
Q_ALU
                                       "SHF/ALU,QK/SHF"
Q_ALU(FRAC)
                                       "SHF/ALU,QK/SHF.FL"
Q_ALU.LEFT
                                      "SHF/LEFT,QK/SHF"
"SHF/ALU.DT,DT/LONG,QK/SHF"
Q_ALU.LEFT3
                                       "QK/SHF,SHF/LEFT3"
                                       "SHF/RIGHT,QK/SHF"
Q_ALU.RIGHT
Q_ALU.RIGHT2
                                       "SHF/RIGHT2,QK/SHF"
                                       "QK/D"
Q...D
Q_D(FRAC)(B)
                                       "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,QK/SHF.FL"
Q_D+KE3
                                       "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/A+B, SHF/ALU, QK/SHF"
Q_D+KE3+1
                                       "RAMX/D, AMX/RANX, KMX/@1, BMX/KMX, ALU/A+B+1, SHF/ALU, QK/SHF"
Q_D+KCJ.LEFT
                                       "RAMX/D, AMX/RANX, KMX/@1, BMX/KMX, ALU/A+B, SHF/LEFT, QK/SHF"
Q_D+LC
                                       "RAMX/D, AMX/RAMX, BMX/LC, ALU/A+B, SHF/ALU, QK/SHF"
                                       "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_D-KED
                                       "RAMX/D,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_D-LC
Q_D - Q
                                      "RAMX/D;AMX/RANX.OXT,DT/@1;ALU/A;SHF/ALU;QK/SHF"
"RAMX/D;AMX/RANX.OXT,DT/@1;KMX/@2;BMX/KMX;ALU/A;B;SHF/LEFT;QK/SHF"
"RAMX/D;AMX/RANX.OXT,DT/@1;BMX/PACKED.FL;ALU/OR;QK/SHF"
Q_D.OXTC3
Q_D.OXTED+KED.LEFT
Q_D.OXTEJ.OR.PACK.FP
Q_D.AND.KEJ
                                       "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/AND, SHF/ALU, QK/SHF"
Q_D.AND.KCJ.RIGHT
Q_D.AND.KCJ.RIGHT2
                                       "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/AND, SHF/RIGHT, QK/SHF"
                                       "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/AND, SHF/RIGHT2, QK/SHF"
                                       "RAMX/D, AMX/RAMX, SPO.R/LOAD.LC, SPO.RC/@1, BMX/LC, ALU/AND, SHF/ALU, QK/SHF"
Q_D.AND.RCEJ
                                       *RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF*
Q_D.ANDNOT.RCE3
Q_D.LEFT3
                                       "RAMX/D, AMX/RAMX, ALU/A, SHF/LEFT3, QK/SHF
Q_D.OR.KED
                                       "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/QR, SHF/ALU, QK/SHF"
```

```
Q_D.OR.RCE3
Q_D.RIGHT
                                                  *RAMX/D,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/OR,SHF/ALU,QK/SHF*
                                                  "RAMX/D, AMX/RAMX, ALU/A, SHF/RIGHT, QK/SHF"
                                                  "RAMX/D,AMX/RAMX,ALU/A,SHF/RIGHT2,QK/SHF"
Q_D.RIGHT2
                                                  "RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
G_D.SXTE3
                                                  "QK/SHF,ALU/XOR,AMX/RAMX,RAMX/D,BMX/RBMX,RBMX/Q,SHF/ALU"
"QK/DEC.CON"
Q_D.XOR.Q
Q_DEC.CON
Q_IB.BDEST
                                                  "IBC/BDEST,QK/ID,MCT/ALLOW.IB.READ"
Q_IB.DATA
                                                  "QK/ID,MCT/ALLOW.IB.READ"
Q_ID(SC)
                                                  "CID/READ.SC,QK/ID"
                                                  "CID/READ.KMX,ID.ADDR/@1,QK/ID"
"KMX/@1,BMX/KMX,ALU/B,SHF/ALU,QK/SHF"
"AMX/RAMX.OXT,DT/LONG,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF"
"KMX/@1,BMX/KMX,ALU/B,SHF/ALU.DT,DT/INST.DEP,QK/SHF"
"KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT,QK/SHF"
QLIDED
Q_KC3
Q_KCJ+1
Q_KE3.CTX
Q_KCJ.RIGHT
                                                  "KMX/@1,BMX/KMX,ALU/B,SHF/RIGHT2,QK/SHF"
Q_KCJ.RIGHT2
                                                  "AMX/LA,ALU/A,SHF/ALU,QK/SHF"

"AMX/LA,KMX/Q1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
QLLA
QLLA+KED
                                                  "AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF"
Q_LA+Q
                                                  "AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"

"AMX/LA,KMX/@1,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"

"AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF"

"AMX/LA,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF"

"BMX/LB,ALU/B,SHF/ALU,QK/SHF"

"BMX/LC,ALU/B,SHF/ALU,QK/SHF"
Q_LA-KEJ
Q_LA.AND.KC3
QLLA.ANDNOT.RCEJ
Q_LB
Q_LC
                                                  "RAMX/Q,AMX/RAMX,ALU/NOTA,SHF/ALU,QK/SHF"
Q.NOT • Q
                                                  "LA_RAC@1],AMX/LA,ALU/NOTA,Q_ALU"
"BMX/PACKED.FL,ALU/B,SHF/ALU,QK/SHF"
Q.NOT.RED
Q._PACK.FP
Q_PC
                                                  "BMX/PC,ALU/B,SHF/ALU,QK/SHF"
                                                  "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,QK/SHF.FL"
"RBMX/Q,BMX/RBMX,ALU/B,SHF/ALU,QK/SHF.FL"
"RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B+1,SHF/ALU,QK/SHF"
Q.Q(FRAC)
Q_Q(FRAC)(B)
Q._Q+D
Q_Q+KED
Q_Q+KC3+1
                                                  RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU,QK/SHF
Q_Q+LC
                                                  "RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,QK/SHF"
Q_Q+PC
Q_Q-D
0-0-0-1
                                                  "RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/A-B-1,SHF/ALU,QK/SHF"
                                                  "RAMX/Q,AMX/RAMX,KBMX/U],BMX/KMX,ALU/A-B-1,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B-1,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,KMX,BMX/LC,ALU/A-B,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B-1,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B-1,SHF/ALU,QK/SHF"
"RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,QK/SHF"
Q.Q~KED
Q_Q-KEJ-1
Q_Q~LC
Q_Q-LC-1
Q_Q-MASK-1
Q_Q.OXTEJ-KEJ
Q_Q.OXTEJ.LEFT
Q_Q.OXTEJ.DR.D
                                                  "RAMX/Q,AMX/RAMX.OXT,DT/@1,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,QK/SHF"
                                                  "RAMX/Q,AMX/RAMX.OXT,DT/@1,ALU/A,SHF/LEFT,QK/SHF"
"RAMX/Q,AMX/RAMX.OXT,DT/@1,RBMX/D,BMX/RBMX,ALU/OR,SHF/ALU,QK/SHF"
Q_Q.AND.KEJ
                                                  "RAMX/Q, AMX/RAMX, KMX/@1, BMX/KMX, ALU/AND, SHF/ALU, QK/SHF
Q.Q.AND.KCJ.RIGHT2
                                                  "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT2,QK/SHF"
                                                  "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SHF/RIGHT,QK/SHF
Q_Q.AND.KEJ.RIGHT
                                                  *RAMX/Q,AMX/RAMX,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/LB,ALU/AND,SHF/ALU,QK/SHF*
Q_Q.AND.RC3
                                                  "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/AND,SHF/ALU,QK/SHF
"RAMX/Q,AMX/RAMX,RBMX/D,BMX/RBMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_Q.AND.RCE3
Q.Q.ANDNOT.D
Q_Q.ANDNOT.KE3
                                                  "RAMX/Q, AMX/RAMX, KMX/@1, BMX/KMX, ALU/ANDNOT, SHF/ALU, QK/SHF
                                                  "RAMX/Q,AMX/RAMX,SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/ANDNOT,SHF/ALU,QK/SHF"
Q_Q.ANDNOT.RCC3
Q_Q.LEFT
                                                  "QK/LEFT
Q_Q.LEFT2
                                                  "QK/LEFT2"
                                                  "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SHF/ALU,QK/SHF"
QLQ.OR.KED
Q_Q.ORNOT.MASK
                                                  "RAMX/Q, AMX/RAMX, BMX/MASK, ALU/ORNOT, SHF/ALU, QK/SHF'
Q.Q.RIGHT
                                                  "QK/RIGHT"
Q_Q.RIGHT2
                                                  "QK/RIGHT2"
Q_Q.SXTC3
                                                  "RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/ALU,QK/SHF"
                                                  "RAMX/Q;AMX/RAMX;KHX/@1;BMX/KMX;ALU/XOR;SHF/ALU;QK/SHF"
"SPO.AC/LOAD.LAB;SPO.ACN/PRN;AMX/LA;RBMX/Q;BMX/RBMX;ALU/ANDNOT;SHF/ALU;QK/SHF"
"SPO.AC/LOAD.LAB;SPO.ACN/PRN+1;AMX/LA;ALU/A;SHF/ALU;QK/SHF"
Q_Q.XOR.KE3
Q.R(PRN).ANDNOT.Q
Q_R(PRN+1)
Q_R(PRN+1).AND.Q
                                                  *SPO.AC/LOAD.LAB,SPO.ACN/PRN+1,AMX/LA,RBMX/Q,BMX/RBMX,ALU/AND,SHF/ALU,QK/SHF*
```

```
Q_R(SC)
                                            "ALU/A,SHF/ALU,AMX/LA,SPO.AC/LOAD.LAB,SPO.ACN/SC,QK/SHF"
Q_R(SRC!1).AND.KEJ
                                            *SPO.AC/LOAD.LAB,SPO.ACN11/SRC.OR.1,AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF*
Q_RC(SC)
                                            "ALU/B,SHF/ALU,BMX/LC,SPO/LOAD.LC.SC,QK/SHF"
                                            "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SHF/ALU,QK/SHF"
Q_RCE3
Q_RCE3(FRAC)
                                            "SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SHF/ALU,QK/SHF.FL"
Q_RC3
                                            "SPO.R/LOAD.LAB.SPO.RAB/@1.AMX/LA.ALU/A.SHF/ALU.QK/SHF"
Q_RE3(FRAC)
                                            "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SHF/ALU,QK/SHF.FL"
Q_REJ.AND.KEJ
                                            *SFO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/AND,SHF/ALU,QK/SHF*
Q_REJ.AND.KEJ.RIGHT
                                            "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/AND,BMX/KMX,KMX/@2,SHF/RIGHT,QK/SHF"
                                            "SFO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,KMX/@2,BMX/KMX,ALU/ANDNOT,SHF/ALU,QK/SHF"
"ALU/OR,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/KMX,KMX/@2,QK/SHF"
Q_REJ.ANDNOT.KEJ
Q_REJ.OR.KEJ
                                            "ALU/B,BMX/KMX,KMX/SC,SHF/ALU,QK/SHF"
Q_SC
Q_SHF
                                            "QK/SHF"
R(DST)_ALU
                                            "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST"
R(DST)_D.SXT[].RIGHT
                                            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST"
"RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SHF/RIGHT,SPO.AC/WRITE.RAB,SPO.ACN11/DST.DST"
R(PRN)_O+D.RLOG
                                            "ALU/A+B.RLOG,BMX/RBMX,RBMX/D,AMX/RAMX.OXT,DT/LONG,R(PRN)_ALU"
R(PRN)_ALU
                                            "SHF/ALU, SPO.AC/WRITE.RAB, SPO.ACN/PRN"
R(PRN)_D
                                            "RAMX/D, AMX/RAMX, ALU/A, SHF/ALU, SPO. AC/WRITE. RAB, SPO. ACN/PRN"
R(PRN)_D+KE3.RLOG
R(PRN)_D-KE3.RLOG
                                            RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R(PRN)..ALU
                                            "RAMX/D, AMX/RAMX, KMX/@1, BMX/KMX, ALU/A-B.RLOG, DT/LONG, R(PRN)_ALU"
R(PRN)_D.OR.Q
R(PRN)_DCJQ
                                            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,R(PRN)_ALU"
                                            "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/@1,R(PRN)_ALU"
R(PRN)_KEJ
R(PRN)_LA+KEJ.RLOG
R(PRN)_LA+Q
R(PRN)_LA+CJ.RLOG
                                            "KMX/Q1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
"AMX/LA,KMX/Q1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R(PRN)_ALU"
"AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B.SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
"AMX/LA,KMX/Q1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R(PRN)_ALU"
R(PRN)_LACIMASK
                                            "AMX/LA,BMX/MASK,ALU/@1,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
                                            "BMX/C,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.AC//PRN"
"BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN"
R(PRN)_LC
R(PRN)_PACK.FP
R(FRN)_Q
R(PRN)_Q+KEJ.RLOG
R(PRN)_Q-KEJ.RLOG
                                            "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/FRN"
                                            "RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/LONG,R(PRN)_ALU"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B.RLOG,DT/LONG,R(PRN)_ALU"
R(PRN+1)_ALU
                                            "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
                                            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO,AC/WRITE.RAB,SPO.ACN/PRN+1"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
R(PRN+1)_D
R(PRN+1)..D.OR.Q
R(PRN+1)_KE3
                                            "KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
R(PRN+1)_LA
                                            "AMX/LA,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1
R(PRN+1)_LC
                                            "BMX/LC,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/PRN+1"
R(PRN+1)_Q
                                            "RAMX/Q, AMX/RAMX, ALU/A, SHF/ALU, SPO. AC/WRITE. RAB, SPO. ACN/PRN+1"
R(SC)_ALU
                                            "SHEZALU,SPO.ACZWRITE.RAB,SPO.ACNZSC"
                                            "RAMX/D;AMX/RAMX;ALU/A;SHF/ALU;SPO.AC/WRITE.RAB;SPO.ACN/SC"
"KMX/Q;BMX/KMX;ALU/B;SHF/ALU;SPO.AC/WRITE.RAB;SPO.ACN/SC"
"AMX/LA;ALU/A;SHF/ALU;SPO.AC/WRITE.RAB;SPO.ACN/SC"
R(SC)_D
R(SC)_KED
R(SC)_LA
R(SC)_LA+D
                                            *AMX/La,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC*
R(SC)_LA-D
R(SC)_LC
                                            "AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SC"
                                            "ALU_LC,R(SC)_ALU"
R(SC)_0
                                            "RAMX/Q, AMX/RAMX, ALU/A, SHF/ALU, SPO. AC/WRITE. RAB, SPO. ACN/SC"
R(SP1)_ALU
                                            "SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
R(SP1)_D
                                            "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
"KMX/@1,BMX/KMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
"BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
R(SP1)_KE3
R(SP1)_PACK.FP
                                            "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1.SP1"
"BMX/LC,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1+1"
"RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN/SP1+1"
R(SP1)_Q
R(SP1+1)_LC
R(SP1+1)_Q
R(SRC!1)_ALU
                                            "SHF/ALU, SFO.AC/WRITE.RAB, SPO.ACN11/SRC.OR.1"
R(SRC!1)_D(B)
R(SRC)_ALU
                                            "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.OR.1"
                                            "SHF/ALU, SPO.AC/WRITE.RAB, SPO.ACN11/SRC.SRC"
```

```
"RAMX/D,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
"RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,DT/WORD,R(SRC)_ALU"
R(SRC)_D
R(SRC)_D(B)
R(SRC)_D+KEJ.RLOG
R(SRC)_D-KEJ.RLOG
R(SRC)_LC
                                                 "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A-B.RLOG,DT/WORD,R(SRC)_ALU"
                                                 "BMX/LC,ALU/B,R(SRC)_ALU"
R(SRC)_Q
                                                 "RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.AC/WRITE.RAB,SPO.ACN11/SRC.SRC"
                                                 "SPO.R/WRITE.RAB,SPO.RAB/R6,RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B.RLOG,SHF/ALU"
"AMX/LA,BMX/KMX,KMX/@1,ALU/A+B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"
"AMX/LA,BMX/KMX,KMX/@1,ALU/A-B.RLOG,DT/WORD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/R6"
R6_D+KE3.RLOG
R6_LA+KEJ.RLOG
R6_LA-KEJ.RLOG
RC(SC)_O-LC
                                                 "ALU_O-LC,RC(SC)_ALU"
RC(SC)_ALU
                                                 "SHF/ALU,SPO/WRITE.RC.SC"
RC(SC)_ALU.RIGHT
                                                 "SPO/WRITE.RC.SC.SHF/RIGHT"
RC(SC)_D
                                                "ALU_D,RC(SC)_ALU"
RC(SC)_Q
                                                "ALU_Q,RC(SC)_ALU"
                                                "RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCC38VA_D+Q
                                                "AMX/RAMX.OXT,DT/LONG,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"AMX/RAMX.OXT,DT/LONG,KMX/@2,BMX/KMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCEJLO
RCEJLO+KEJ+1
                                                "AMX/RAMX.OXT,DT/LONG,BMX/LC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
RCC3_0+LC+1
                                                "AMX/RAMX.OXT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/ALU,SF0.R/WRITE.RC,SP0.RC/@1"
"AMX/RAMX.OXT,DT/LONG,BMX/MASK,ALU/A+B+1,SHF/RIGHT2,SP0.R/WRITE.RC,SP0.RC/@1"
RCEJ_O+MASK+
RCEJ_O+MASK+1.RIGHT2
                                                "AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCCJ_O-D
                                                "SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"SHF/LEFT,SPO.R/WRITE.RC,SPO.RC/@1"
RCEJLALU
RCEJLALU.LEFT
RCC3_ALU.LEFT2
                                                "SPO.R/WRITE.RC,SPO.RC/@1,SHF/ALU.DT,DT/LONG"
RCCD_ALU.LEFT3
                                                "SPO.R/WRITE.RC,SPO.RC/@1,SHF/LEFT3"
RCCJ_ALU.RIGHT
                                                "SHF/RIGHT,SPO.R/WRITE.RC,SPO.RC/@1"
                                                "SHF/RIGHT2, SPO.R/WRITE.RC, SPO.RC/@1"
RCCJ_ALU.RIGHT2
                                                "RAMX/D, AMX/RAMX, ALU/A, SHF/ALU, SPO.R/WRITE.RC, SPO.RC/@1
RCE 1...D
RCEJ_D(B)
                                                "RBMX/D,BMX/RBMX,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCC3_D+KC3
                                                "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
                                                "RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"RAMX/D,AMX/RAMX,OXT,DT/@2,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"RAMX/D,AMX/RAMX,BMX/KMX,KMX/@2,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"RAMX/D,AMX/RAMX,BMX/MASK,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCEJ_D-KEJ
RCEJ_D.OXTEJ
RCCJ_D.AND.KCJ
RCEILD.AND.MASK
RCEILD.ANDNOT.Q
                                                *RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/ANDNOT,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1*
RCED_D.CTX
                                                "RAMX/D,AMX/RAMX,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/01"
RCCJ_D.LEFT
RCCJ_D.LEFT3
                                                "RAMX/D, AMX/RAMX, ALU/A, SHF/LEFT, SPO.R/WRITE.RC, SPO.RC/@1"
"RAMX/D, AMX/RAMX, ALU/A, SHF/LEFT3, SPO.R/WRITE.RC, SPO.RC/@1"
                                                "RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"SPO.RC/@1,SPO.R/WRITE.RC,ALU/ORNOT,AMX/RAMX,RAMX/D,BMX/KMX,KMX/@2,SHF/ALU"
RCCJ_D.OR.KCJ
RCCJ_D.OR.Q
RCCJ_D.ORNOT.KCJ
                                                "RAMX/D;AMX/RAMX.SXT;DT/@2;ALU/A;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
"KMX/@2;BMX/KMX;ALU/B;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
"AMX/RAMX.OXT;DT/LONG;KMX/@2;BMX/KMX;ALU/A+B+1;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
RCCD_D.SXTCD
RCCO_KCO
RCCJ_KCJ+1
RCCOLKCO.LEFT2
                                                "KMX/@2,BMX/KMX,ALU/B,SHF/ALU.DT,DT/LONG,SPO.R/WRITE.RC,SPO.RC/@1"
                                                "KMX/02,BMX/KMX,ALU/B,SHF/LEFT3,SPO.R/WRITE.RC,SPO.RC/01"
"KMX/02,BMX/KMX,ALU/B,SHF/RIGHT2,SPO.R/WRITE.RC,SPO.RC/01"
RCCJ_KCJ.LEFT3
RCCJ_KCJ.RIGHT2
RCEDILLA
                                                "AMX/LA,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCC3_LA+LB.CTX
                                                *AMX/LA,BMX/LB,ALU/A+B,SHF/ALU.DT,DT/INST.DEP,SFO.R/WRITE.RC,SFO.RC/@1*
RCC3...LA-KC3
                                                *AMX/LA,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU,SFO.R/WRITE.RC,SPO.RC/@1
RCCJ_LA.AND.KCJ
                                                "ALU_LA.AND.KC@23,RCC@13_ALU"
                                                "AMX/LA,ALU/A,SHF/ALU.DT,DT/INST.DEP,SPO.R/WRITE.RC,SPO.RC/@1"
"BMX/LB,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCEJLLA.CTX
RCEJLLB
                                                "BMX/LB,ALU/B,SHF/LEFT,SPO.R/WRITE.RC,SFO.RC/@1"
RCCULLB.LEFT
                                                "BMX/LC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1
"RAMX/Q,AMX/RAMX,ALU/NOTA,RC[@1]_ALU"
RCC3_NOT.Q
RCCJ_PACK+FP
                                                *BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SFO.RC/@1*
                                                "BMX/PC,ALU/B,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
"RAMX/Q,AMX/RAMX,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/@1"
RCEDLPC
RCCI_Q
RCCJ_Q+1
                                                "ALU_O+Q+1,RCE@13_ALU"
```

```
"RAMX/G.AMX/RAMX,BMX/KMX,KMX/02.ALU/A+B.SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
"ALU/A+B.RAMX/G.AMX/RAMX,BMX/LC,SPO.R/WRITE.RC,SPO.RC/01"
"RAMX/G.AMX/RAMX,BMX/PC,ALU/A+B.SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
"RAMX/G.AMX/RAMX,BMX/PC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
"RAMX/G.AMX/RAMX,BMX/PC,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
"ALU/A-B.RAMX/G.AMX/RAMX,BMX/LC,SPO.R/WRITE.RC,SPO.RC/01"
 RCC3_Q+KC3
RCC3_Q+LC
RCC3_Q+PC
 RCCJ_Q+PC+1
RCCJ_Q-KCJ
RCCJ_Q-LC
                                                                                                                   "RAMX/Q,AMX/RAMX,BMX/MASK,ALU/A-B-1,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
"RAMX/Q,AMX/RAMX.OXT,DT/02,ALU/A,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
"RAMX/Q,AMX/RAMX,BMX/KMX,KMX/02,ALU/AND,SHF/ALU,SPO.R/WRITE.RC,SPO.RC/01"
 RCC3_Q-MASK-1
RCC3_Q.OXTC3
 RCC3..Q.AND.KC3
                                                                                                                  "RAMX/Q;AMX/RAMX;BMX/KMX;KMX/@2;ALU/AND;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
"RAMX/Q;AMX/RAMX;BMX/KMX;KMX/@2;ALU/ANDNOT;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
"RAMX/Q;AMX/RAMX;ALU/A;SHF/LEFT3;SPO.R/WRITE.RC;SPO.RC/@1"
"RAMX/Q;AMX/RAMX;ALU/A;SHF/LEFT3;SPO.R/WRITE.RC;SPO.RC/@1"
"RAMX/Q;AMX/RAMX;ALU/A;SHF/RIGHT;SPO.R/WRITE.RC;SPO.RC/@1"
"ALU_Q;SHF/RIGHT2;SFO.R/WRITE.RC;SPO.RC/@1"
"RAMX/Q;AMX/RAMX.SXT;DT/@2;ALU/A;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
"BMX/Q;AMX/RAMX.SXT;DT/@2;ALU/A;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
"BMX/Q;AMX/RAMX.SXT;DT/@2;ALU/A;SHF/ALU;SPO.R/WRITE.RC;SPO.RC/@1"
 RCCJ_Q.ANDNOT.KCJ
RCCJ_Q.LEFT
 RCCJ_Q.LEFT3
RCCJ_Q.RIGHT
 RCCJ_Q.RIGHT2
 RCC3_Q.SXTC3
 RCEJ_RLOG.RIGHT
 REJ&VA_LA+KEJ
                                                                                                                                   La,KMX/@2,BMX/KMX,ALU/A+B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1
 REJ8VA_LA-KEJ
REJ8VA_LA-KEJ.RLOG
                                                                                                                   *AMX/La,KMX/02,BMX/KMX,ALU/A-B,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01*
*AMX/La,KMX/02,BMX/KMX,ALU/A-B.RLOG,DT/LONG,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01*
                                                                                                                  "AMX/LA,KMX/@2,BMX/KMX,ALU/A-B.RLOG,DT/LONG,VAK/LOAD,SHF/ALU,SPO.R/WRITE.RAB,SPO.R
"RAMX/Q,AMX/RAMX,KMX/@2,BMX/KMX,ALU/A-B,VAK/LOAD,SPO.R/WRITE.RAB,SPO.RAB/@1"
"SPO.R/WRITE.RAB,SPO.RAB/@1,AMX/RAMX,OXT,DT/LONG,ALU/A,SHF/ALU"
"AMX/RAMX.OXT,DT/LONG,BMX/LB,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/RAMX.OXT,DT/LONG,BMX/KMX,KMX/.1,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/RAMX.OXT,DT/LONG,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/RAMX.OXT,DT/LONG,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/LEFT"
"SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/LEFT"
"SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/LEFT"
"SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/LEFT"
"SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"
"SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1,SHF/RIGHT2"
 RCJ&VA_Q~KCJ
 REJ_0
REJ_0+LB+1
 REJ..0-1
 RCJ_O-D
 REJ_O-KEJ
 REJ_O-LB
  REJ_O-Q
 REJ_ALU
 REJ_ALU.LEFT
 RCJ_ALU.LEFT3
RCJ_ALU.RIGHT
                                                                                                                  "SPO.R/WRITE.RAB,SPO.RAB/01.FMF/RIGHT2"

"SPO.R/WRITE.RAB,SPO.RAB/01.FAMX/D,AMX/RAMX,ALU/A,SHF/ALU"

"SPO.R/WRITE.RAB,SPO.RAB/01.FAMX/D,AMX/RAMX,KMY.02.BMX/KMX,ALU/A+B,SHF/ALU"

"SPO.R/WRITE.RAB,SPO.RAB/01.FAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU"

"SPO.R/WRITE.RAB,SPO.RAB/01.FAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU"

"SPO.R/WRITE.RAB,SPO.RAB/01.FAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,SHF/ALU"
 RCJ_ALU.RIGHT2
 RCJ_D
 RCJ_D+KCJ
 RED_D+Q
RCJ_D+Q+1
RCJ_D-KCJ
                                                                                                                   "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,KMX/@2,BMX/KMX,ALU/A-B,SHF/ALU"
 RCJ_D-LC-1
                                                                                                                    ALU_D-LC-1,RC@13_ALU'
                                                                                                                  "SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/AND,AMX/RAMX,RAMX/D,BMX/KMX,KMX/@2,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/OR,AMX/RAMX,RAMX/D,BMX/LC,SHF/ALU"
RCJ_D-Q
RCJ_D.AND.KCJ
REJ_D.OR.LC
REJ_D.OR.PACK.FP
REJ_D.OR.Q
                                                                                                                   "SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/OR,AMX/RAMX,RAMX/D,BMX/PACKED.FL,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/OR,SHF/ALU"
                                                                                                                   "BMX/KMX,KMX/02,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
"SPO.R/WRITE.RAB,SPO.RAB/01,AMX/LA,ALU/A,SHF/ALU"
"AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
REJ_KEJ
REJ_LA
 RCJ_LA+D
                                                                                                                "AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,RBMX/D,BMX/RBMX,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,BMX/KMX,KMX/@2,ALU/A+B+1,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,BMX/KMX,KMX/@2,ALU/A+B+1,RCG]J_ALU"
"AMX/LA,BMX/KMX,KMX/@2,ALU/A+B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,BMX/LC,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,BMX/LO,BMX/RBMX,ALU/A+B+1,RCG1J_ALU"
"AMX/LA,RBMX/G,BMX/RBMX,ALU/A+B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/LA,BMX/KMX,KMX/@2,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
REJ_LA+D+1
REJ_LA+KEJ
REJ_LA+KEJ+1
REJ_LA+KEJ•RLOG
RCJ_LA+LC
RCJ_LA+MASK+1
RCJ_LA+Q
REJ_LA-D
REJ_LA-KEJ
 REJ_LA-KEJ.RLOG
                                                                                                                   "AMX/La,BMX/KMX,KMX/@2,ALU/a-B.RLOG,DT/LONG,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
                                                                                                                  "ALU/A-B-1,AMX/MASK,SPD.R/WRITE.RAB,SPD.RAB/01,SHF/ALU"
"AMX/LA,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
"AMX/LA,RBMX/Q,BMX/RBMX,ALU/A-B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
"AMX/LA,BMX/KMX,KMX/02-ALU/AND,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
"AMX/LA,BMX/D,BMX/RBMX,ALU/OR,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
"AMX/LA,BMX/MASK,ALU/ORNOT,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
"BMX/LB,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
REJ_LA-MASK-1
REJ_LA-Q
REJ_LA.AND.KEJ
REJ_LA.OR.D
REJ_LA.ORNOT.MASK
RED_LB
```

```
"BMX/LC,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/@1"
"BMX/LC,ALU/B,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"
"AMX/RAMX.OXT,DT/LONG,ALU/NOTA,RC@1]_ALU"
 RCI_LC
RCI_LC.RIGHT
 RE3_NOT.0
                                                                                            "RAMX/D,AMX/RAMX,ALU/NOTA,RC@1]_ALU"
"BMX/MASK,AMX/RAMX.OXT,DT/LONG,ALU/ORNOT,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01"
  RED_NOT.D
 RED_NOT.MASK
                                                                                            *RAMX/O.AMX/RAMX.ALU/NOTA.RCG1J_ALU*
*BMX/PACKED.FL,ALU/B,SHF/ALU,SPO.R/WRITE.RAB,SPO.RAB/01*
*SPO.R/WRITE.RAB,SPO.RAB/01,RAMX/Q.AMX/RAMX,ALU/A,SHF/ALU*
  RED_NOT.Q
 RCJ_PACK.FP
                                                                                            "ALU_O+Q+1,RCQ1]_ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,ALU/A+B+1,BMX/KMX,KMX/.4,AMX/RAMX,RAMX/Q,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/A+B,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1,RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,SHF/ALU"
 RCJ_Q+1
RCJ_Q+5
RCJ_Q+KCJ
RCJ_Q+LB
 RCJ_Q+LC
RCJ_Q-D
                                                                                            "SPO.R/WRITE.RAB.SPO.RAB/@1.ALU/A-B-1.AMX/RAMX.RAMX/Q.BMX/RBMX.RBMX/D.SHF/ALU"
"SPO.R/WRITE.RAB.SPO.RAB/@1.RAMX/Q.AMX/RAMX.BMX/KMX/W2.ALU/A-B.SHF/ALU"
"RAMX/Q.AMX/RAMX.BMX/KMX/KMX/W2.ALU/A-B.RLOG.DT/LONG.SHF/ALU.SPO.R/WRITE.RAB.SPO.RAB/@1"
 RED..Q-D-1
 RCJ_Q-KCJ
RCJ_Q-KCJ.RLOG
                                                                                           "SPO.R/WRITE.RAB,SPO.RAB/@1.RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU"
"ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/@1.RAMX/Q,AMX/RAMX,BMX/LC,ALU/A-B,SHF/ALU"
"ALU/AND,SPO.R/WRITE.RAB,SPO.RAB/@1.ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@2"
"SPO.R/WRITE.RAB,SPO.RAB/@1.ALU/ANDNOT,AMX/RAMX,RAMX/Q,BMX/KMX,KMX/@2,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1.ALU/ANDNOT,AMX/RAMX,BMX/WAMX,RAMX/B,SHF/ALU"
"SPO.R/WRITE.RAB,SPO.RAB/@1.RAMX/Q,AMX/RAMX,BMX/KMX,KMX/@2,ALU/ORNOT,SHF/ALU"
"ALU_Q,SHF/RIGHT,SPO.R/WRITE.RAB,SPO.RAB/@1"
 RC3_0-LC
 REJ_Q.AND.KEJ
REJ_Q.ANDNOT.KEJ
 REJ_Q.OR.D
 RCJ_Q.ORNOT.KCJ
RCJ_Q.RIGHT.1
                                                                                            BMX/O, MSC/READ.RLOG, ALU/B, SHF/RIGHT, SPO.R/WRITE.RAB, SPO.RAB/01
 RCJ_RLOG.RIGHT.1
 SC&STATE_STATE-REJ(EXP)
                                                                                             ·LAB_RC@1],AMX/LA,EBMX/AMX.EXF,MSC/LOAD.STATE,EALU/A-B,SMX/EALU,SCK/LOAD.
                                                                                            "AMX/RAMX.OXT,DT/LONG,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
"BMX/KMX,KMX,G1,AMX/RAMX.OXT,DT/LONG,ALU/A-B,SMX/ALU,SCK/LOAD"
"SMX/ALU,SCK/LOAD"
 SC_O-KEJ
SC_ALU
 SC_ALU(EXP)
                                                                                             SMX/ALU.EXP,SCK/LOAD
                                                                                            "RAMX/D,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD"
"RAMX/D,AMX/RAMX,ALU/A,SMX/ALU,EXP,SCK/LOAD"
"RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU,SCK/LOAD"
 SC_D(EXP)
SC_D(EXP)(A)
                                                                                            "RRMX/D;AMX/RAMX;EBMX/AMX-EXF;EALL/B;SHX/EALL;SLK/LUAD"
"RRMX/D;AMX/RAMX;ALU/B;SMX/ALU,EXF;SCK/LUAD"
"RAMX/D;AMX/RAMX;KMX/Q1;BMX/KMX;ALU/A-B;SMX/ALU;SCK/LUAD"
"RAMX/D;AMX/RAMX.OXT;DT/Q1;FMX/Q2;BMX/KMX;ALU/A-B;SMX/ALU;SCK/LUAD"
"RAMX/D;AMX/RAMX;OXT;DT/Q1;FMX/KMX;KMX;Q2;ALU/XOR;SC_ALU"
"RAMX/D;AMX/RAMX;KMX/Q1;BMX/KMX;ALU/AND;SMX/ALU;SCK/LUAD"
 SC_D(EXP)(B)
SC_D-KEJ
SC_D.OXTEJ-KEJ
 SC_D.OXTEJ.XOR.KEJ
SC_D.AND.KEJ
                                                                                          "RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/OR,SMX/ALU,SCK/LOAD"
"RAMX/D,AMX/RAMX.SXT,DT/@1,ALU/A,SMX/ALU,SCK/LOAD"
"SMX/EALU,SCK/LOAD"
"KMX/@1,EMX/KMX,EALU/B,SMX/EALU,SCK/LOAD"
"KMX/@1,EMX/KMX,EALU/B,SMX/ALU,SCK/LOAD"
"KMX/@1,EMX/KMX,ALU/B,SMX/ALU,SCK/LOAD"
"AMX/LA,ALU/A,SMX/ALU,SCK/LOAD"
"AMX/LA,KMX/@1,BMX/KMX,ALU/AND,SMX/ALU,SCK/LOAD"
"BMX/LC,ALU/B,SMX/ALU,EXP,SCK/LOAD"
"BMX/LC,ALU/B,SMX/ALU,EXP,SCK/LOAD"
"BMX/LC,ALU/B,SMX/ALU,EXP,SCK/LOAD"
 SC_D.OR.KCJ
 SC_D.SXTCJ
SC_EALU
 SCLFE
SCLKEJ
 SC_KEJ.ALU
 SC_LA
SCLLA.AND.KEJ
SCLLC(EXP)
SC_NABS(SC-FE)
                                                                                           "EBMX/FE, EALU/NABS.A-B, SMX/EALU, SCK/LOAD"
                                                                                           "SMX/EALU,EBMX/KMX,SCK/LOAD,KMX/.F,EALU/B"
"RAMX/Q,AMX/RAMX,ALU/A,SMX/ALU,SCK/LOAD"
 SC_PSLADDR
SC_Q
SC_Q(EXP)
                                                                                           "RAMX/Q,AMX/RAMX,EBMX/AMX.EXF,EALU/B,SMX/EALU,SCK/LOAD"
                                                                                          "RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,SMX/EALU-SCK/LOAD"
"RBMX/Q,BMX/RBMX,ALU/B,SMX/ALU.EXP,SCK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/KMX.EMX/@1,ALU/A+B;SMX/ALU,SCK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/KMX.EMX/@1,ALU/A+B;SMX/ALU,SCK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/KMX.EMX/@1,ALU/AND,SMX/ALU,SCK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/KMX.EMX/@1,ALU/AR,SMX/ALU,SCK/LOAD"
"RAMX/Q,AMX/RAMX.SXT,DT/@1,ALU/A,SMX/ALU,SCK/LOAD"
"SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SMX/ALU,SCK/LOAD"
"SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,SMX/ALU,SCK/LOAD"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A;SMX/ALU,SCK/LOAD"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A;SMX/ALU,SCK/LOAD"
SC_Q(EXP)(B)
SC_Q+KEJ
SC_Q-KED
SC_Q.AND.KEJ
SC_Q.OR.KEJ
SCLQ.SXTE3
SCLRCE3
SCLRCE3(EXP)
SC_RED
SC_REJ(EXP)
SC_REJ.AND.KEJ
                                                                                           "SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,SMX/ALU.EXP,SCK/LOAD"
"ALU/AND,AMX/LA,SPO.R/LOAD.LAB,SPO.RAB/@1,BMX/KMX,KMX/@2,SMX/ALU,SCK/LOAD"
                                                                                           "EALU/A+1,SMX/EALU,SCK/LOAD"
SC_SC+1
```

VA_D+LC

```
SC_SC+EXP(Q)(A)
                                                 "EALU/A+B,EBMX/AMX.EXP,SMX/EALU,SCK/LOAD,AMX/RAMX,RAMX/Q"
SC_SC+FE
                                                 "EBMX/FE, EALU/A+B, SMX/EALU, SCK/LOAD"
SC_SC+KED
                                                 "KMX/Q1,EBMX/KMX,EALU/A+B,SMX/EALU,SCK/LOAD"
                                                 "EALU/A+B,EBMX/SHF,VAL,SMX/EALU,SCK/LOAD"
"EBMX/FE,EALU/A-B,SMX/EALU,SCK/LOAD"
"KMX/@1,EBMX/KMX,EALU/A-B,SMX/EALU,SCK/LOAD"
"EBMX/SHF,VAL,EALU/A-B,SMX/EALU,SCK/LOAD"
"EBMX/FE,EALU/ANDNOT,SMX/EALU,SCK/LOAD"
SC_SC+SHF.VAL
SC_SC-FE
SC_SC-K[]
SC_SC-SHF.VAL
SC_SC.ANDNOT.FE
SC_SC.ANDNOT.KEJ
SC_SC.OR.KEJ
SC_SHF.VAL
                                                 "KMX/@1,EBMX/KMX,EALU/ANDNOT,SMX/EALU,SCK/LOAD"
                                                 "KMX/@1,EBMX/KMX,EALU/OR,SMX/EALU,SCK/LOAD"
"EBMX/SHF.VAL,EALU/B,SMX/EALU,SCK/LOAD"
SC_STATE
                                                 "EALU/A, MSC/LOAD.STATE, SMX/EALU, SCK/LOAD"
SC_STATE.ANDNOT.KEJ
SC_STATE.OR.KEJ
                                                 *EALU/ANDNOT,EBMX/KMX,MSC/LOAD.STATE,SMX/EALU,SCK/LOAD,KMX/@1*
                                                 "EALU/OR, EBMX/KMX, MSC/LOAD. STATE, SMX/EALU, SCK/LOAD, KMX/@1"
SD_NOT.SD
                                                 "SGN/NOT.SD"
                                                 "SGN/SD.FROM.SS"
"SGN/CLR.SD+SS"
SD_SS
SS_08SD_0
SS_ALU15
                                                 "SGN/LOAD.SS"
SS_SD
SS_SS.XOR.ALU15&SD_ALU15
                                                 "SGN/SS.FROM.SD"
                                                 "SGN/SS.XOR.ALU"
                                                 "AMX/RAMX.OXT,DT/LONG,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
STATE_O(A)
STATE_AMX.EXP
STATE_D(EXP)
                                                 "EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
"RAMX/D,AMX/RAMX,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
"EBMX/FE,EALU/B,MSC/LOAD.STATE"
STATE_FE
                                                 "STATE_K[ZERO]"
"STATE_K[.1]"
"STATE_K[.3]"
STATE_FIRST
STATE_INNEROBJ
STATE_INNERSRC
                                                                                                             JEDITPC STATES
JMATCHC STATES
                                                 "KMX/@1,EBMX/KMX,EALU/B,MSC/LOAD.STATE"
"STATE_KCZERO]"
STATE_KED
STATE_OUTER
STATE_PREDEC
                                                 "STATE_K[.80]"
                                                 "RAMX/Q,AMX/RAMX,EBMX/AMX.EXP,EALU/B,MSC/LOAD.STATE"
STATE_Q(EXP)
STATE_SC.VIA.KMX
                                                 "MSC/LOAD.STATE, EALU/B, EBMX/KMX, KMX/SC
STATE_SKPLONG
                                                 "STATE_K[.4]"
                                                                                                             JSKPC STATES
                                                 "EALU/A+1,MSC/LOAD.STATE"
"EBMX/FE,EALU/A+B,MSC/LOAD.STATE"
"KMX/01,EBMX/KMX,EALU/A+B,MSC/LOAD.STATE"
STATE_STATE+1
STATE_STATE+FE
STATE_STATE+KCJ
STATE_STATE-FE
STATE_STATE-KCJ
                                                 "EBMX/FE,EALU/A-B,MSC/LOAD,STATE"
"KMX/@1,EBMX/KMX,EALU/A-B,MSC/LOAD.STATE"
                                                 "STATE_STATE.ANDNOT.K[.4]"
STATE_STATE.AN.SKPLONG
                                                "STATE_STATE.ANDNOT.KC.3FJ"
"STATE_STATE.ANDNOT.KC.7FJ"
STATE .. STATE . AN . 5TOO
STATE_STATE.AN.6TO4
STATE_STATE.AN.DESTDBL
                                                 "STATE_STATE.ANDNOT.KC.63"
STATE_STATE.AN.NOTPREDEC
                                                 "STATE_STATE.ANDNOT.KE.7FJ"
STATE_STATE.AN.PREDECZERO
                                                 "STATE_STATE.ANDNOT.KC.COJ"
STATE_STATE.ANDNOT.FE
                                                 "EBMX/FE,EALU/ANDNOT,MSC/LOAD.STATE"
                                                "KMX/@1,EBMX/KMX,EALU/ANDNOT,MSC/LOAD.STATE"
"MSC/LOAD.STATE,EBMX/SHF.VAL,EALU/ANDNOT"
"EALU/OR,EBMX/FE,MSC/LOAD.STATE"
"KMX/@1,EBMX/KMX,EALU/OR,MSC/LOAD.STATE"
"STATE_STATE.OR.KL.3]"
"STATE_STATE.OR.KL.4]"
STATE_STATE.ANDNOT.KCJ
STATE_STATE.ANDNOT.SHF.VAL
STATE_STATE.OR.FE
STATE_STATE.OR.KCJ
STATE_STATE.OR.ADJINP
STATE_STATE.OR.DEST
                                                 "STATE_STATE.OR.KE.63"
"STATE_STATE.OR.KE.73"
STATE_STATE.OR.DESTDBL
STATE_STATE.OR.FILL
STATE_STATE.OR.FLOAT
                                                 "STATE_STATE.OR.KE.603"
STATE_STATE.OR.MOVE
STATE_STATE.OR.PATT1
                                                 "STATE_STATE.OR.K[.50]"
                                                 "STATE_STATE.DR.KE.13"
STATE_STATE.OR.PATT2
                                                 "STATE_STATE.OR.KE.23"
SWAPD
                                                 "DK/BYTE.SWAP"
VA_ALU
                                                "VAK/LOAD"
VA_D
VA_D+K[]
                                                "RAMX/D,AMX/RAMX,ALU/A,VAK/LOAD"
"RAMX/D,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD"
```

"RAMX/D,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD"

PCK/VA+4

```
VA_D+Q
VA_D.OXTE3+Q
VA_D.ANDNOT.KE3
VA..KE3
VA_LA
VA_LA+D
VALLA+KE3
VA_LA+KEJ+1
VA_LA+PC
VA...LA+Q
VA..LA-D
VA_LA-KEJ
VA_LA-KEJ-1
VA.LA-Q
VA_LA.AND.LC
VA_LA.ANDNOT.KEJ
VA_LB+D.OXT
VALPC.
VA..Q
VA_Q+D
VA_Q+KC3
VA_Q+LB
VA_Q+LB.FC
VA_Q+LC
VA_Q+PC
VA_Q-KED
VA..Q--LB
VA_Q.ANDNOT.KEJ
VA_RCEJ
VA_REJ
VA_VA+4
```

```
"RAMX/D,AMX/RAMX,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD"
"RAMX/D,AMX/RAMX.OXT,DT/@1,BMX/RBMX,ALU/A+B,VAK/LOAD"
"RAMX/D,AMX/RAMX,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD"
"KMX/@1,BMX/KMX,ALU/B,VAK/LOAD"
"AMX/LA,ALU/A,VAK/LOAD"

"AMX/LA,RBMX/I,BMX/RBMX,ALU/A+B,VAK/LOAD"

"AMX/LA,BMX/KMX,KMX/@1,ALU/A+B,VAK/LOAD"

"AMX/LA,BMX/KMX,KMX/@1,ALU/A+B+1,VAK/LOAD"
"AMX/LA,BMX/PC,ALU/A+B,VAK/LOAD"
"AMX/LA,RBMX/Q,BMX/RBMX,ALU/A+B,VAK/LOAD"
"AMX/LA,RBMX/D,BMX/RBMX,ALU/A-B,VAK/LOAD"
"AMX/LA,BMX/KMX,KMX/@1,ALU/A-B,VAK/LOAD"
"AMX/LA,BMX/KMX,KMX/@1,ALU/A-B-1,VAK/LOAD"
"VAK/LOAD,ALU/A-B,AMX/LA,BMX/RBMX,RBMX/Q,SHF/ALU"
"AMX/LA,BMX/LC,ALU/AND,VAK/LOAD"
"AMX/LA,BMX/KMX,KMX/@1,ALU/ANDNOT,VAK/LOAD"
"BMX/LB,ALU/A+B,AMX/RAMX.OXT,DT/BYTE,VAK/LOAD"
"BMX/PC,ALU/B,VAK/LOAD"
"RAMX/Q, AMX/RAMX, ALU/A, VAK/LOAD"
"VAK/LOAD, ALU/A+B, AMX/RAMX, BMX/RBMX, RAMX/Q, RBMX/D, SHF/ALU"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/A+B,VAK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/LB,ALU/A+B,VAK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/PC,OR,LB,ALU/A+B,VAK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/LC,ALU/A+B,VAK/LOAD"
"RAMX/Q,AMX/RAMX,BMX/PC,ALU/A+B,VAK/LOAD"
"RAMX/Q,AMX/RAMX,KMX/Q1,BMX/KMX,ALU/A-B,VAK/LOAD"
"RAMX/Q, AMX/RAMX, BMX/LB, ALU/A-B, VAK/LOAD"
"RAMX/Q,AMX/RAMX,KMX/@1,BMX/KMX,ALU/ANDNOT,VAK/LOAD"
"SPO.R/LOAD.LC,SPO.RC/@1,BMX/LC,ALU/B,VAK/LOAD"
"SPO.R/LOAD.LAB,SPO.RAB/@1,AMX/LA,ALU/A,VAK/LOAD"
```

"SUB/RET,J/20"

RETURN20

```
. TOC
                 'Macro definition
                                                                : Non-transfer macros*
B.FORK
                                                                  "LAB_R(SP1),QK/ID,CLR.IB.COND,PC_PC+N,SUB/SPEC,J/B.FORK"
                                                                  *DT/BYTE*
BYTE
                                                                  "SUB/SPEC, J/C.FORK"
"MCT/INVALIDATE, VAK/NOP"
"SUB/CALL"
CAFORK
CACHE . INVALIDATE
CALL
                                                                  "CALL,J/@1"
"MSC/CHK.FLT.OPR"
"MSC/CHK.ODD.ADDR"
CHK.FLT.OPR
CHK.ODD.ADDR
CLK.UBCC
                                                                  *CCK/LOAD.UBCC*
CLR.FPD
                                                                  "MSC/CLR.FPD"
                                                                  "IBC/CLR.1-5.COND"
"IBC/CLR.0,IEK/ISTR"
"IBC/CLR.1"
CLR.IB.COND
CLR.IB.OPC
CLR.IB.SPEC
                                                                  'IBC/CLR.0.1,IEK/ISTR'
'IBC/CLR.0-3'
'IBC/CLR.2-3'
'IBC/CLR.1-5.COND'
'MSC/CLR.NEST.ERR'
'SGN/CLR.SD+SS'
CLR.IBO-1
CLR.IBO-3
                                                                                                                   #DISCARD -11 INSTR & OPERAND
#11 MODE DISCARD ISTREAM OPERAND
#2ND PART OF Q/D IMMEDIATE
CLR. IB2-3
CLR. IB2-5
CLR.NEST.ERR
CLR.SD&SS
                                                                  "SUB/SPEC,J/E.FORK"
"IEK/EACK"
E.FORK
EXCEPT.ACK
FLUSH. IB
                                                                  "IRC/FLUSH, VAK/LOAD, IEK/ISTR"
G.FORK
                                                                  "SUB/SPEC.J/G.FORK"
INHIBIT. IB
                                                                  "HCT/MEN.NOP"
                                                                 "MCI/MEN.NUF"
'IEK/IACK"
'IEK/IACK"
'IEK/ISTR'
'IRDO,CLK.UBCC,IRD1,SUB/SPEC,J/A.FORK'
'IRDO,CLK.UBCC,IRD1,SUB/SPEC,J/A.FORK'
'IRDO,CLK.UBCC,IRD1,SUB/SPEC,J/A.FORK'
'LA_R(SP1)&LB_R(SRC),D_LB.PC,VAK/LOAD,Q_IB.DATA,SC_KE.101,PCK/PC+N,MSC/IRD,SUB/SPEC,J/DPO'
'LA_R(SP2)&LB_R(SP1),D&VA_LB,SC_ALU(EXP),FE_LA(EXP),SS_ALU15'
'MSC/IRD,QK/ID,MCT/ALLOW.IB.READ,IBC/CLR.1-5.COND,PCK/PC+N'
INTRPT.ACK
INTRPT.STROBE
IRD
IRD.11
IRDO
IRD1
                                                                  "MSC/LOAD.ACC.CC"
"VAK/NOP.MCT/READ.V.NEWPC"
"VAK/NOP.MCT/READ.V.NEWPC"
"DT/LONG"
LOAD.ACC.CC
LOAD.IB
LOAD.IB.11
LONG
MEMORY . NOP
                                                                  "HCT/HEM.NOP"
                                                                  "SI/MUL+,SC_SC-K[.1],BEN/MUL"
"SI/MUL-,SC_SC-K[.1],BEN/MUL"
"D_D.RIGHT2,SI/MUL",INTRPT.STROBE"
MUL . OXT
MUL.1XT
MULH.DONE
MULP.DONE
                                                                  *D_D.RIGHT2,SI/MUL+,INTRPT.STROBE*
POLY. DONE
                                                                  "ACF/CONTROL, ACM/POLY. DONE"
RETURNO
                                                                  "SUB/RET,J/O"
                                                                  SUB/RET,J/1°
SUB/RET,J/10°
SUB/RET,J/100°
SUB/RET,J/10C°
SUB/RET,J/10E°
RETURN1
RETURN10
RETURN100
RETURN10C
RETURN10E
                                                                  SUB/RET,J/10E
SUB/RET,J/12*
SUB/RET,J/18*
SUB/RET,J/1F*
RETURN12
RETURN18
RETURN1F
RETURN2
```

WRITE.G.DEST

```
"SUB/RET,J/24"
"SUB/RET,J/3"
"SUB/RET,J/4"
RETURN24
RETURNS
RETURNS
RETURN40
                                            "SUB/RET,J/40"
                                            SUB/RET, J/60*
SUB/RET, J/61*
RETURN60
RETURN61
                                             SUB/RET, J/8
RETURNS
                                            "SUB/RET,J/9"
"SUB/RET,J/OF"
RETURN9
RETURNE
                                             "SUB/RET,J/81"
RETURNED
SET.CC(BYTE)
                                            "CCK/INST.DEP,DT/BYTE"
SET.CC(INST)
SET.CC(LONG)
                                            "CCK/INST.DEP.DT/INST.DEP"
"CCK/INST.DEP.DT/LONG"
                                            *CCK/ROR*
*CCK/INST.DEP,DT/WORD*
*MSC/SET.FPD*
SET.CC(ROR)
SET.CC(WORD)
SET.FPD
SET.NEST.ERR
                                            "MSC/SET.NEST.ERR"
SET.PSL.C(AMX)
                                             "CCK/C_AMXO"
SET.V
                                            "CCK/SET.V"
SPEC
                                            "LAB_R(SP1), Q_IB.DATA, CLR.IB.COND, PC_PC+N, MCT/ALLOW.IB.READ, SUB/SPEC, J/C.FORK"
                                            "LAB_R(SP1),Q_IB.DATA,CLR.IB.COND,PC_PC+N,MCT/ALLOW.IB.READ,SUB/SPEC,J/G.FORK"
"IBC/START"
SPECG
START. IB
                                            "IBC/STOP"
STOP. IB
                                            "MCT/TEST.RCHK,VAK/NOP"
"MCT/TEST.WCHK,VAK/NOP"
TEST.TB.RCHK
TEST.TB.WCHK
                                            "ACF/TRAP,ACM/@1"
TRAP.ACCED
WORD
                                            'DT/WORD'
                                            "LAB_R(SP1), QK/ID, CLR.IB.COND, PC_PC+N.SUB/SPEC, J/WRD"
"LAB_R(SP1), QK/ID, CLR.IB.COND, PC_PC+N, SUB/SPEC, J/WRG"
WRITE.DEST
```

•TOC *Macro definition	: Branch enable macros*	
AC.LOW?	"BEN/INTERRUPT" #,J3/3"	
	"BEN/ACCEL" #, J3/3"	
ACC.SYNC?		
ACCEL?	"BEN/ACCEL"	
ALIGNED?	"BEN/TB.TEST" #, J5/17"	
ALU.N?	"BEN/ALU" #, J4/07"	
ALU1-07	BEN/ALU1-0	
ALU?	"BEN/ALU"	
BCDSGN?	"BEN/DECIMAL" #,J2/2"	
C31?	"BEN/C31"	
CONSOLE.MODE?	"BEN/PSL.MODE" #, J5/1B"	
D(1)?	"BEN/MUL"	
D.BO?	"BEN/D.BYTES" , J4/OE"	
D.B1?	"BEN/D.BYTES" ,J4/OD"	
D.B27	"BEN/D.BYTES" J4/OB"	
D.BYTES?	"BEN/D.BYTES"	
D.NE.0?	"BEN/SIGNS" + J3/5" PREFERED FORM	
DO?	"BEN/D3-0" #,J4/0E"	
D2-07	"BEN/D3-0"	
D27	"BEN/D3-0" J4/OB"	
D3-0?	"BEN/D3-0"	
D31?	"BEN/SIGNS" , J3/6"	
D3?	"BEN/D3-0" #,J4/07"	
DATA.TYPE?	"BEN/DATA.TYPE"	
DBL?	"BEN/DATA.TYPE"	
EALU.N?	"BEN/EALU"	
EALU.Z?	"BEN/EALU" ;,J4/OB"	
EALU?	"BEN/EALU"	
END.DP1?	BEN/END.DP1	
FPD?	"BEN/LAST.REF" \$,J4/07"	
IB.TEST?	"BEN/IB.TEST"	
INT?	"BEN/INTERRUPT"	
INTERRUPT.REG?	"BEN/INTERRUPT" #,J3/5"	
IRO.C31?	"BEN/ALU"	
IRO?	"REN/ALU"	
IR1?	"BEN/IR2-1" /, J3/6"	
IR2-1?	"BEN/IR2-1"	
LAST.REF?	*BEN/LAST.REF*	
	ACTIVITY OF THE PROPERTY OF TH	
MODE.LSS.ASTLVL?	"BEN/REI"	
MUL?	"BEN/MUL"	
NEST.ERR?	"BEN/LAST.REF"),J4/OB"	
NEO I VENN.	ELIVEROTURE! TOWN OF	
FC.MODES?	"BEN/PC.MODES"	
PSL+C?	"BEN/PSL.CC" ,J4/0E"	
PSL.CC?	"BEN/PSL.CC"	
PSL.MODE?	"BEN/PSL.MODE"	
PSL.N?	"BEN/PSL.CC" J4/7"	
PSL.V?	"BEN/PSL.CC" /, J4/OD"	
PSL.Z?	"BEN/PSL.CC" J4/0B"	
PTE.VALID?	"BEN/TB.TEST" #, J5/OF"	
Q31?	"BEN/SIGNS" ;,J3/3"	
QUAD?	"BEN/DATA.TYPE"	

RLOG.EMPTY? ROR?	BEN/ALU1-0° BEN/ROR°	;,J4/7°
SC.GT.O? SC.NE.O? SC?	"BEN/SC" "BEN/MUL" "BEN/SC"	;,J3/3°
SIGNS? SRC.PC? SS? STATE(7)?	"BEN/SIGNS" "BEN/SRC.PC" "BEN/EALU" "STATE7-4?"	#COMP MODE, BEN ON SRC R = PC #,J4/0E"
STATEO? STATE1-0? STATE1?	"BEN/STATE3-0" "BEN/STATE3-0" "BEN/STATE3-0"	;,J4/0E" ;,J4/0C" ;,J4/0D"
STATE2? STATE3-0? STATE3? STATE4?	"BEN/STATE3-0" "BEN/STATE3-0" "BEN/STATE3-0" "BEN/STATE3-0"	;,J4/08" ;,J4/07"
STATES? STATE6? STATE7-4?	"BEN/STATE7-4" "BEN/STATE7-4" "BEN/STATE7-4"	
TB.TEST?	"BEN/TB.TEST"	
VA31-30? VA31?	"BEN/PSL.MODE" "BEN/PSL.MODE"	;,J5/07" ;,J5/0F"
Z? ZONED?	"BEN/Z" "BEN/DECIMAL"	**J2/1*
.BIN	≯MAKE LISTING ROOM FOR BINARY F	ROM HERE ON

	·		

APPENDIX B

SAMPLE MICROPROGRAM FOR SYSTEM REVISION > 7

This appendix contains a sample VAX 11/780 microprogram, which performs an unsigned binary search on a vector of longwords in main memory. The parameters of the routine, the value to be searched for and the beginning and end of the vector, are passed in registers.

A command file that assembles, loads, and executes this sample microprogram is provided in the VAX 11/780 WCS kit. To invoke this file in the VMS environment, type:

@[SYSEXE]WCSTOLTST

This command file assembles the input listing (Section B.1) and produces the listing file (Section B.2) and the object file (Section B.3) which are written to [VAXWCSTOL]SAMPLE.MCR and [VAXWCSTOL]SAMPLE.ULD. It then loads the object file into the extended WCS and runs the test program BSTEST (Appendix D). BSTEST executes an XFC instruction, which causes the sample microprogram loaded in the WCS to be executed. If the microprogram executes properly, BSTEST prints the following message on the terminal:

"Successful Test Completion"

B.1 THE INPUT FILE (.MIC)

```
.TOC 'Binary search routine'
.REGION /1C00:1FFF
.BOUNDS/BSERCH:1C00,1FFF
.BOUNDS/BSERCH:1C00,1FF
.BO
```

SRCH:		; ;GET UPPER BOUND ADDR TO Q ;INITIALIZE STATE REGISTER
	p_RCROJ	; ;GET COMPARAND TO HOLD IN RC
		#PREPARE TO WRITE COMPARAND TO RC#WRITE COMPARAND, GET LOWER BOUND
SRCH.1:	; Q_(LA+Q).RIGHT, INTRPT.STROBE, STATEO?	; \$COMPUTE MIDPOINT ADDRESS \$TEST FOR INTERRUPT REQUESTS \$IS IT TIME TO STOP?
™O SRCH.2:	Q_Q.ANDNOT.KE.3J, VA_ALU, LC_RCET1J,	### ### ### ### ######################
	ALII KEZERNI.	<pre> \$STATEO=1. SEARCH FAILED. NO MATCH. \$RETURN Z=1 TO FLAG FAILURE. \$MOVE ON TO THE NEXT INSTRUCTION \$</pre>
=110 SRCH.3:		FREADY MIDPOINT ENTRY OF VECTOR COMPARE MIDPOINT EQL UPPER BOUND
	#111	-}INTERRUPT REQUEST IS UP ;TAKE IT. RESUME FROM REG'S AS IS.

```
; WE HAVE ALSO SET THE MICROBRACH Z BIT ACCORDING TO A COMPARE OF

# THE MEMORY ADDRESS WITH THE CURRENT UPPER BOUND. IF THEY ARE
# EQUAL, THIS IS THE LAST POSSIBLE COMPARISON. A MATCH FAILURE
# HERE IMPLIES THAT THERE IS NO MATCH TO BE FOUND.
        ALU_D-LC, $COMPARE MEMORY TO COMPARAND LONG,CLK.UBCC, $RECORD COMPARE RESULT $LATCH LOWER BOUND INTO LA CL
                                FLATCH LOWER BOUND INTO LA (LB HAS 777
FIS MIDPOINT EQL UPPER BOUND?
        ;0-----;ALU Z=0. NOT END OF SEARCH
ALU?,J/SRCH.5 ;TEST RESULT OF COMPARE
::::O
        ALU7, J/SRCH.5
        #1-----#ALU Z=1. END OF SEARCH
        STATE_KC.1], #SET STATEO TO MARK END OF SEARCH.
                                 #CHECK FOR LAST CHANCE MATCH
        ALU?
==1010
J/SRCH.6
        Q_Q-K[.4], $UPPER LIMIT MUST BE LESS THAN THIS RER2]_ALU, $REMEMBER IN R2
                                 ₹GO TRY AGAIN
        J/SRCH.1
RCR13_Q, FOUND IT!
CCK/NZ_ALU,VC_O,LONG, SET Z=O TO INDICATE MATCH
CLR.IB.OPC,PC_PC+1, FGO TO NEXT INSTRUCTION
        J/IRD
Q._(Q+LB).RIGHT, COMPUTE NEW MXDPOINT, LOOP
        INTRPT.STROBE,
        STATEO?, J/SRCH. 2
                                 #CHECK FOR END: LOOP
; DEFINE LABLES TO INTERFACE WITH PCS 0062: IRD:
        INT.B:
04F8:
```

B.2 THE LISTING FILE (.MCR)

ş	NEWSA	M.MCR	MI	CRO2 1L(02) 18-JAN-82 16:15:06
9			Ta	ble of Contents
ŷ	2	Machine definition	:	Control word chart
ş	56	Machine definition	:	ACF, ACM, ADS, ALU, AMX
ŷ	97	Machine definition	:	BEN, BMX
ŷ	150	Machine definition	:	CCK, CID, DK, DT
ŷ	205	Machine definition	:	EALU, EBMX, FEK, FS, IEK, IBC
ŷ	255	Machine definition	:	ID.ADDR, J
9	330	Machine definition	:	KMX
ŷ	405	Machine definition	:	MCT, MSC
ŷ	452	Machine definition	:	PCK, QK, RAMX, RBMX
ŷ	487	Machine definition	:	SCK, SGN, SHF, SI, SMX
ŷ	529	Machine definition	:	SPO, SPO.AC, SPO.ACN, SPO.ACN11, SPO.R
ŷ	568	Machine definition	:	SPO.RAB, SPO.RC, SUB, VAK
ş	617	Machine definition	:	Validity checks
ŷ	624	Macro definition	:	Register transfer macros
ŷ	1538	Macro definition	:	Non-transfer macros
ø	1634	Macro definition	:	Branch enable macros
á	1770	Disamu coasab soutise		

NEWSAM.MCR # VAXDEF.MIC

MICRO2 1L(02) 18-JAN-82 16:15:06

Page 2

#1 .NOLIST

#Inhibit listing for VAXDEF.MIC

```
# NEWSAM.MCR
                      MICRO2 1L(02)
                                              18-JAN-82 16:15:06
                                                                                                                                                  Pase
# BSERCH.MIC
                      Binary search routine
                                                            .TOC "Binary search routine"
                                      11729
                                                            .REGION /1COO,1FFF
.BOUNDS/BSERCH:1COO,1FFF
                                      $1730
$1731
                                                                                                        #User was space.
#This defines the report boundries
                                      1732
                                                                                                        for the U-code microword summary page
                                      11733
                                                                                                        ;and names the report boundary BSERCH.
                                      11734
                                      11735
                                                 # Sample microcode to perform an unsigned binary search through
                                      11736
                                                 ; a vector of aligned longwords in main memory.
                                      11737
                                      1738
                                                 # INPUTS
                                      1739
                                                           RO - Search comparand. Routine succeeds by finding a
                                     $1740
$1741
$1742
$1743
                                                                      memory cell containing same data as RO.
                                                R1 - Lower address bound. Alianed lonaword address of lowest address of vector to be searched.

R2 - Upper address bound. Alianed lonaword address of highest address of vector to be searched.

It is implied that R1 lssu R2, and that the memory between the
                                      1744
                                      11745
                                      1746
                                                   addresses in R1 and R2 contains a sorted vector, in ascending
                                      $1747
                                                   unsidned order.
                                      £1748
                                     $1749
$1750
                                                   Outputs if search finds a match.
                                                           CC<Z> - Clear
RO - Search comparand.
                                     11751

    Match address. Address of longword containing same data as RC.
    Used by search for temporary address values.

                                      11752
                                                           F: 1
                                      11753
                                                           R2
                                     $1754
$1755
                                                   Outputs if search does not find a match.
                                     11756
                                                           CC<Z> - Set
                                      11757
                                                                   - Search comparand.
                                                           RO

Used by search for temporary address values.
Used by search for temporary address values.

                                      11758
                                                           R1
                                      £1759
                                                           R2
                                      £1760
```

-	NEWSAN BSERCH		MICRO2 1 Binary se	LL(02) 1 earch routi		2 16:15:06	Pase	40
				#1761 #1762	SRCH:		-•	
u	1004,	0000,003C,19C0,FA10	,1404,7C05	11763		Q_RCR23,	∮GET UPPER BOUND ADDR TO Q ∮INITIALIZE STATE REGISTER	
				\$1766	•	;	-	
U	1005,	0800,003C,0180,FA00	,0000,1008	#1768		DRCRO]	#GET COMPARAND TO HOLD IN RC	
	1000-	0001,003C,0180,FR08	-0000-1009	#1769 #1770 #1771		ALU_D,	.) ;PREPARE TO WRITE COMPARAND TO RC ;WRITE COMPARAND, GET LOWER BOUND	
L,	10007	0001700327018077208	,0000,100,	#1772 #1773		CAPUNIANCE II JUNEO	TWATTE COM MARKEY OF LOWER BOOKE	
				#1774	SRCH.1:	;		
				\$1775		Q_(LA+Q).RIGHT,	COMPUTE MIDPOINT ADDRESS	
		0050 4744 0450 5000		\$1776 \$1777			ITEST FOR INTERRUPT REQUESTS	
U	1009,	005C,1714,01CO,F800	4000,1000	#1/// #1778		STATEO?	FIS IT TIME TO STOP?	
					=0			
						:0	STATEO=0. KEEP LOOKING FOR MATCH.	
				\$1781	J		FORCE LONGWORD ALIGNMENT	
				1782			JGET READY TO READY MIDPOINT OF VECTOR	3
				£1783			FLATCH COMPARAND INTO LC	•
IJ	1C00,	0019,2E24,0DC0,F908	0200,1006	11784			IS THERE AN INTERRUPT REQUEST?	
				11785				
				11786			JSTATEO=1. SEARCH FAILED. NO MATCH.	
				#1787		ALU_KCZEROJ,	;	
				#1788			FRETURN Z=1 TO FLAG FAILURE.	
				#1789		CLR.IB.OPC,PC_PC+1,	MOVE ON TO THE NEXT INSTRUCTION	
U	1001,	C018,0038,1980,F804	4050,0062	#1790		J/IRD	;	
				11791				
					=110	****	AND THESPANDS ACRESTS	
						;110		
				\$1794 11705			FREADY MIDPOINT ENTRY OF VECTOR	
	1004	0010,0020,0180,4210,	AA1A. 1000	₹1795 ₹1796		ALU_RER23.XOR.Q, CLK.UBCC,J/SRCH.4	COMPARE MIDFOINT EQL UPPER BOUND	
t.J	1000	0010,0020,0180,4210,	001011000	#1796 #1797		CLN.UBCC1J/BRCH.4	•	
				#1798		1111	JINTERRUPT REQUEST IS UP	
U	1C07.	0000,003C,0180,F800,	0000,04FB	\$179 9			FTAKE IT. RESUME FROM REG'S AS IS.	

	NEWS BSEI			MICRO2 10 Binary sea			16:15:06		Pase	41
					\$1800 \$1801 \$1802 \$1803 \$1804	F THE ME	MORY ADDRESS WITH THE CL	CH Z BIT ACCORDING TO A COMPARE O PRRENT UPPER BOUND. IF THEY ARE BLE COMPARISON. A MATCH FAILURE MATCH TO BE FOUND.	F	
					11805	SRCH. 41	;	- 1		
					\$1806		ALU_D-LC,	JCOMPARE MEMORY TO COMPARAND		
					\$1807		LONG, CLK, UBCC,	FRECORD COMPARE RESULT		
					11808			FLATCH LOWER BOUND INTO LA (LB H	AS 777	
ı	J 100	C . 0	011,0100,0180,F888,0	010,1002	11809		Z?	FIS MIDPOINT EQL UPPER BOUND?		
		"			#1810					
					11811	=0	; 0	- JALU Z=O. NOT END OF SEARCH		
ı	J 100	2, 0	000,1B3C,0180,F800,0	0000,1C0A	£1812			FTEST RESULT OF COMPARE		
					11813	•				
					11814		<u> </u>	-#ALU Z=1. END OF SEARCH		
					11815	:	STATE_K[.1],	#SET STATEO TO MARK END OF SEARC	н.	
- (J 100	3, 0	000,1B3C,0580,F800,	L404,7COA	#1816		ALU?	#CHECK FOR LAST CHANCE MATCH		
					#1817					
					;1818	=1010				
					11819	SRCH.5:	\$1010	-#ALU Z=0, C=1. RO GTRU MEM		
					11820	(Q_Q+KE.4J,	#LOWER LIMIT MUST BE GREATER THA	N THIS	
					11821	1	RER1J_ALU,	FREMEMBER IN R1.		
ı	J 100	4, 0	019,2014,11CO,FA88,	0000,1COD	11822	•	J/SRCH.6	;		
					11823					
					11824		1011	-JALU Z=0, C=0. RO LSSU MEM		
					11825		Q_Q-KE.43,	PUPPER LIMIT MUST BE LESS THAN T	HIS	
					11826		RCR23_ALU,	FREMEMBER IN R2		
ı	J 1C01	B, ()	019,2000,11C0,FA90,0	0000,1009	#1827 #1828		J/SRCH.1	FGO TRY AGAIN		
					#1829			-FALU Z=1, C=1. RO EQL MEM		
					;1830	1	RCR1J_Q,	FOUND IT!		
					;1831			FSET Z=0 TO INDICATE MATCH		
					11832			#GO TO NEXT INSTRUCTION		
ı	J 1CO	· C	001,203C,0180,FA8C,4	1050,0062	11833		J/IRD			
					11834					
					11835					
					11836		Q_(Q+LB).RIGHT,	COMPUTE NEW MIDPOINT, LOOP		
			0AD - 771 A - 01 CO - E000	1000-1000	#1837 #1838		INTRPT.STROBE, STATEO?,J/SRCH.2	CHECK FOR END, LOOP		
ı	. 1601	J, 0	04D,3714,01C0,F800,4	10001100	#1839					
					£1840		LABLES TO INTERFACE WIT	TH PCS		
					11841		IRD:			
					11842	04F8:	INT.B:			

; NEWSAM.MCR ;	MICRO2 1L(02) 18-JAN-82 16:15:06 Cross Reference Listins - Field Names and Defined Values	Pase	42
J INT.B IRD SRCH SRCH.1 SRCH.2 SRCH.3 SRCH.4 SRCH.5 SRCH.6	326 \$ 1799		

∮ NEWSAM÷MCR		(02)	18-JAN-82 16:15:06 Listins - Macro Names
,			
AC.LOW?	1636 #		
ACC.SYNC?	1637 #		
ACCEL?	1638 #		
ALIGNED?	1639 #		
ALU.N?	1640 #		
ALU1-07	1641 # 1642 #	1812	1816
ALU?	626 #	1017	1010
ALU1 ALU_0(A)	627 #		
ALU_0+D	628 #		
ALU_0+D+1	629 #		
ALULO+KCI	630 #		
ALU_0+KCJ+1	631 #		
ALU_O+LB+1	632 #		
ALU_0+LC	633 #		
ALU_0+LC+1	634 #		
ALU_O+MASK+1	635 #		
ALU_O+R	636 #		
ALU_0+Q+1	637 #		
ALU_O-D	638 #		
ALU_0-D-1	639 #		
ALU_O-KC3	640 #		
ALU_O-KEJ-1	641 # 642 #		•
ALU_O-LB	643 #		•
ALU_O-LC ALU_O-LC-1	644 #		
ALU_0-Q	645 #		
ALU_0-Q-1	646 #		
ALU_OCID	647 #		
ALU_OE ILC	648 #		
ALU_D	649 #	1770	
ALU_D(B)	650 #		
ALU_D+KEJ	651 #		
ALU_D+KCJ+1	652 #		
ALU_D+KEJ.RLOG	653 #		
ALU_D+LB	654 #		
ALU_D+LC	655 # 656 #		
ALU_D+LC+1 ALU_D+LC+PSL+C	657 #		
ALU_D+Q	658 #		
ALU_D+Q+1	659 ≢		
ALU_D+Q+PSL+C	660 #		
ALU_D+RLOG	661 #		
ALU_D-KCJ	662 #		
ALU_D-KEJ-1	663 *		
ALU_D-LB	664 #		
ALU_D-LB.RLOG	665 #		
ALU_D-LC	666 #	1806	
ALU_D-LC-1	667 #		
ALU_D-Q	668 #		
ALU_D-Q-1	669 #		
ALU_D.OXTED	670 # 671 #		
ALULD.OXTED+KED ALULD.OXTED+LC	672 #		
ALU_D.OXTE3+Q	673 #		
Calman my + A V + P m 1 / 1/2	W/W #		

P NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06 Cross Reference Listins - Macro Names
ALU_D.OXTCJ-KCJ	674 ‡
ALU_D.OXTEJ-Q	675 #
ALU_D.OXTEJ.AND.KEJ	676 ‡
ALU_D.OXTEJ.ANDNOT.KEJ	677 #
ALU_D.OXTEJ.OR.Q	678 #
ALU_D.AND.KC3	679 #
ALU_D.AND.MASK	680 #
ALU_D.ANDNOT.KC]	681 #
ALU_D.ANDNOT.MASK	682 #
ALU_D.ANDNOT.Q	683 # 684 #
ALU_D.OR.KCJ ALU_D.OR.LC	685 #
ALU_D.OR.Q	686 #
ALU_D.OR.RCEJ	487 #
ALUD.ORNOT.MASK	688 #
ALU_D.SXTEJ	689 #
ALU_D.SXTEJ+KCJ	690 #
ALU_D.SXTCJ+Q	691 #
ALU_D.SXTEJ.AND.KEJ	693 #
ALU_D.SXTEJ.ANDNOT.KEJ	692 #
ALU_D.XOR.KE3	694 #
ALU_D.XOR.LC	695 #
ALUD.XOR.Q	696 #
ALU_D.XOR.RCE3	697 #
ALU_D.XOR.RC3	698 #
ALU_DCJKCJ	699 ‡ 700 ‡
ALU_DEJLB ALU_DEJLC	700 #
ALU_DCJQ	702 #
ALU_KEJ	703 # 1787
ALU_LA	704 #
ALU_LA+K[]	705 #
ALU_LA+K[]+1	706 #
ALU_LA+K[].RLOG	707 #
ALU_LA+LB	708 #
ALU_LA+LC	709 #
ALU_LA+LC+1 ALU_LA+LC+FSL+C	710 # 711 #
ALU_LA+Q	712 #
ALU_LA-D	713 #
ALU_LA-D-1	714 #
ALU_LA-KEJ	715 #
ALU_LA-K[]-1	716 #
ALU_LA-KE3.RLOG	717 #
ALU_LA-LC	718 #
ALU_LA-Q	719 #
ALU_LA-Q-1	720 #
ALU_LA.AND.KCJ	721 #
ALU_LA.AND.LC	722 * 723 *
ALU_LA.ANDNOT.KEJ ALU_LA.ANDNOT.MASK	723 * 724 *
ALU_LA.OR.KEJ	725 #
ALU_LA.XOR.LC	726 #
ALU_LACID	727 #
ALU_LACJLB	728 #

# NEWSAM.MCR	MICRO2 1L(O2) 18-JAN-82 16:15:06 Cross Reference Listins - Macro Names
ALU_LACJQ	729 #
ALU_LB	730 #
ALULIC	731 #
ALU_NOT.D	732 #
ALU_NOT.KEJ	733 #
ALU_NOT+RCC3	734 #
ALU_PACK.FP	735 #
ALU_PC	736 #
ALU_Q	737 #
ALU_Q(B)	738 #
ALU_Q+KCJ	739 #
ALU_Q+KEJ+1	740 #
ALU_Q+LB	741 #
ALU_Q+LB+1	742 #
ALU_Q+LC	743 #
ALU_Q+LC+1	744 #
ALU_Q+LC+PSL.C	745 #
ALU_Q+MASK	746 #
ALU_Q-D	747 #
ALU_Q-D-1	748 #
ALU_Q-KC3	749 #
ALU_Q-LB	750 #
ALU_Q-LC	751 #
ALU_Q-MASK-1	752 #
ALU_Q.OXTE3	753 #
ALU_Q.OXTEJ+D	754 #
ALU_Q.OXTEJ+D+1	755 #
ALU.Q.OXTE3+KE3	756 #
ALU_Q.OXTEJ-D	757 #
ALU_Q.OXTCJ-KCJ	758 #
ALU_Q.OXTCJ.ANDNOT.KCJ	759 #
ALU_Q.OXT[].OR.D	761 #
ALU_Q.OXT[].OR.K[]	760 #
ALU_Q.AND.D	762 #
ALU_Q.AND.KCJ	763 #
ALU_Q.ANDNOT.KC3	764 #
ALU_Q.ANDNOT.MASK	765 #
ALU_Q.ANDNOT.RE3	766 #
ALU_Q.OR.KEJ	767 #
ALU_G.OR.LC	768 #
ALU_Q.ORNOT.KE3	769 #
ALU_Q.SXTC]	770 * 771 *
ALU_Q.SXTEJ+KEJ	771 * 772 *
ALU_Q.SXTCJ+LB ALU_Q.SXTCJ+LB+1	772 * 773 *
	773 * 77 4
ALU_Q.SXTCJ+PC ALU_Q.SXTCJ.ANDNOT.KCJ	774 * 775 *
ALU_Q.XOR.D	770 * 776 *
ALU_Q.XOR.KE3	7/0 * 777 *
ALU_Q.XOR.LC	778 #
ALU_Q.XOR.RCC3	779 #
ALU_QCJD	780 #
ALU_R(DST)	781 #
ALULR(SC).ANDNOT.KC3	782 #
ALU_R(SP1)+KEJ.RLOG	783 *
ringerijul & / III is III in Wu	

Pase 46

ALU_RC(SC)	F NEWSAM.MCR	MICRO2 1L(02)	18-JAN-82 16:15:06
ALU.RCI) 786	ş	Cross Reference L	isting - Macro Names
ALU.RCI) 786	ALL DOZOON	794 +	
ALU_RIOS			
ALU_RE]			
ALU_REJ_AND_KEJ 789			
ALU_RE].AND.KE] 789			
ALU.REJ.ANB.LC ALU.REJ.ANBONT.KEJ ALU.REJ.ANBONT.KEJ ALU.REJ.OR.NEJ ALU.REJ.OR.NEJ ALU.REJ.OR.NEJ ALU.REJ.XOR.Q ALU.REJ.XOR.Q B.FORK BCDSGNY 1644 * BCDSGNY 1644 * BCDSGNY 1644 * BYYE 1541 * C.FORK 1543 * C.GACHE.J.DT CACHE.J.DT CACHE.J.DT CACHE.J.DT CACHE.J.D 799 * CACHE.DJ.LK 803 * CACHE.DJ.LK 803 * CACHE.DJ.LK 804 * CACHE.DEJ. NOCHK CALL 1545 * CALL 1546 * CHK.ODD.ADDR 1548 * CLK.UBCC 1549 * 1796 1807 CLR.FPD 1551 * CLR.IB.OPC 1553 * 1789 1832 CLR.IB.OPC 1553 * CLR.IB.OPC 1554 * CLR.IB.OPC 1555 * CLR.IB.OPC 1556 * CLR.IB.OPC 1557 * CLR.IB.OPC 1558 * CLR.IB.OPC 1559 * CLR.IB.OPC 1559 * CLR.IB.OPC 1559 * CLR.IB.OPC 1549 * CLR.IB.OPC 1559 * CLR.IB.OPC 1549 * CLR.IB.OPC 1559 * CLR.IB.OPC 1549 * CLR.IB.OPC 1559 * CL			
ALU_REJ.ANDNOT.KCJ ALU_REJ.ANDNOT.MASK ALU_REJ.ORNOT.KCJ ALU_REJ.ORNOT.KCJ ALU_REJ.XOR.KCJ ALU_REJ.XOR.KCJ ALU_REJ.XOR.RCJ ALU_REJ.XOR.RCJ ALU_REJ.XOR.RCJ ALU_REJ.XOR.RCJ B.FORK BCDSGM? 1644 * BCDSGM? 1644 * BCDSGM? 1644 * BCDSGM? 1644 * BCPSGMR 1543 * C.FORK C31? CACHE.INVALIDATE 1543 * CACHE.INVALIDATE 1544 * CACHE.P_DCJ CACHE.DCJ CACHE.D			
ALU_REJ.GN.KI] ALU_REJ.GN.KI] ALU_REJ.GN.KI] ALU_REJ.GN.KI] ALU_REJ.GN.KI] ALU_REJ.XOR.G ALU_REJ.XOR.G B.FORK BCDSGM? 1540			
ALU_REJ.OR.KEJ 793			
ALU_RIJ.XOR.KIJ 795			
ALU_RIJ.XOR.RIJ ALU_RIJ.XOR.Q ALU_RIJ.XOR.Q B.FORK B.FORK B.FORK B.FORK BYTE 1541		794 #	
ALU_RCJ.XOR.Q		795 #	
B.FORK BCDSGN7 1644 # BCDSGN7 1644 # BYTE 1541 # C.FORK 1543 # C.GACHE.INVALIDATE CACHE.INVALIDATE CACHE.INVALIDATE CACHE.P.DEJ CACHE.DCJ CACHE.CC CACHE.DCJ CACHE.CC CACHE.DCJ CACHE.CC CACHE.DCJ CACHE.CC CA		796 # 1795	
BCDSGN? BYTE C.FORK C.FORK C31? CACHE.INVALIDATE CACHE.P_DCJ CACHE_DCJ CACHE_DCQUAD) CACHE_D.INST.DEP CACHE_DIJ.NOCHK CALL CACHE_DCJ.NOCHK CALL			
BYTE C.FORK 1543			
C.FORK C31? 1646 # CACHE.INVALIDATE CACHE.P_DC] 798 # CACHE.DCD 799 # CACHE.DC			
C31?			
CACHE.INVALIDATE CACHE.P_DDJ CACHE.DCD CACHELD.QUAD) CACHELD.INST.DEP CACHE.DDJ.LK CACHE.DDJ.LK CACHE.DDJ.LK CACHE.DDJ.LK CACHE.DDJ.LK CACHE.DDJ.NOCHK CALL CACHE.DDJ.NOCHK CACHE.DDJ.NOCH CACHE.DDJ.NOCH CACHE.DDJ.NOCH CACHE.DDJ.NOCH CACHE.DDJ.NOCH			
CACHE.P_DET 798			
CACHECJD			
CACHE_D(QUAD)		799 #	
CACHE_DIJ		800 #	
CACHE_DCJ_LK CACHE_DCJ_NOCHK CACHE_DCJ_PC CACHE_DCJ_NOCHK CACH			
CACHE_DID.NOCHK CALL CALL CALL CALL CALL CALL CALL CAL			
CALL I 1545	CACHELDEJ.LK		
CALLC]	CACHELDED:NOCHK		
CHK.FLT.OPR CHK.ODD.ADDR CHK.ODD.ADDR 1548	CALL		
CHK.ODD.ADDR CLR.UBCC CLR.FPD CLR.IB.COND CLR.IB.OPC CLR.IB.OPC CLR.IB.SPEC CLR.IBO-1 CLR.IBO-3 CLR.IBO-3 CLR.IBO-5 CLR.IBO-5 CLR.IBO-5 CLR.NEST.ERR CLR.SD2SS CONSOLE.MODE? D2Q.D+Q D2VAD+LC D2VAD+C D2VALB D2VALB D2VALB D2VACLR.DC D2VACLR.DC B15 # D2VACLR.DC B16 # D2VACLR.DC B17 # D2VACLR.DC B18 # D2VACLR.DC B19 # D2VACLR.DC B11 # D2VACLR.DC B12 # D2VACLR.DC B13 # D2VACLR.DC B14 # D2VACLR.DC B15 # D2VACLR.DC B15 # D2VACLR.DC B16 # D2VACLR.DC B17 # D2VACLR.DC B18 # D2VACLR.DC B19 # D2VACLR.DC B10 # D2VACLR.DC B11 # D2VACLR.DC B12 # D2VACLR.DC B13 # D2VACLR.DC B14 # D2VACLR.DC B15 # D2VACLR.DC B15 # D2VACLR.DC B16 # D2VACLR.DC B17 # D2VACLR.DC B18 # D2VACLR.DC B19 # D2VACL.DC B19 # B	CALLES	1546 #	
CLR.UBCC 1549 # 1796 1807 CLR.FPD 1551 # CLR.IB.COND 1552 # CLR.IB.OPC 1553 # 1789 1832 CLR.IB.SPEC 1554 # CLR.IBO-1 1555 # CLR.IBO-3 1556 # CLR.IBC-3 1558 # CLR.IBC-3 1559 # CLR.IBC-5 1558 # CLR.SD2S 1560 # CONSOLE.MODE? 1647 # D&Q_D+Q 806 # D&RCCI_PC 807 # D&VA_ALU 808 # D&VA_D+LC 809 # D&VA_D+C 811 # D&VA_LB 811 # D&VA_LB 813 # D&VA_C+LB,PC 815 # D&VA_C+LB,PC 8	CHK.FLT.OPR		
CLR.FPD 1551			
CLR.IB.OPC	CLK.UBCC	1549 # 1796	1807
CLR.IB.SPC	CLR.FPD		
CLR.IB.SPEC 1554 \$ CLR.IBO-1 1555 \$ CLR.IBO-3 1556 \$ CLR.IB2-3 1557 \$ CLR.IB2-5 1558 \$ CLR.NEST.ERR 1559 \$ CLR.SD2SS 1560 \$ CONSOLE.MODE? 1647 \$ D2Q_D+Q 806 \$ D2VA_ALU 808 \$ D2VA_ALU 809 \$ D2VA_D+C 807 \$ D2VA_D+C 811 \$ D2VA_D+C 811 \$ D2VA_D+C 811 \$ D2VA_LB 811 \$ D2VA_LB 813 \$ D2VA_LB 813 \$ D2VA_CLB 814 \$ D2VA_CLB 815 \$ D2VA_CLB 815 \$ D2VA_CLB 816 \$ D2VA_CLB 817 \$ D2VA_CLB 818 \$ D2VA_CLB 818 \$ D2VA_CLB 818 \$ D2VA_CLB 818 \$ D2VA_CLB 819 \$			
CLR.IBO-1	CLR.IB.OFC		1832
CLR.IB0-3 CLR.IB2-3 CLR.IB2-3 CLR.IB2-5 CLR.NEST.ERR CLR.SD2SS CONSOLE.MODE? D2C_D+Q D2C_D+Q D2VA_ALU D2VA_D+LC D2VA_D+Q D2VA_D+Q D2VA_D+Q D2VA_LB D2VA_CB D3VA_CB D3V			
CLR.IB2-3 CLR.IB2-5 CLR.NEST.ERR CLR.SDRSS CONSOLE.MODE? DRQD+Q DRCCI_PC DRVAALU DRVAD+LC DRVAD+C			
CLR.IB2-5 CLR.NEST.ERR CLR.SDASS CLR.SDASS CONSOLE.MODE? D&QD+Q D&QD+Q D&VAALU D&VAALU D&VAD+LC D&VAD+C D&VAD+C D&VAD+C D&VAD+C D&VAD+C D&VAD+C D&VAD+C D&VAD+C D&VALA B11 # D&VALA B12 # D&VALB B13 # D&VALB B13 # D&VALB CAN CONSOLE CON			
CLR.NEST.ERR 1559 # CLR.SD&SS 1560 # CONSOLE.MODE? 1647 # D&Q.D+Q 806 # D&RCEJ_PC 807 # D&VA.ALU 808 # D&VA.D+LC 809 # D&VA.D+C 810 # D&VA.D+C 811 # D&VA.LA 812 # D&VA.LB 813 # D&VA.LB 813 # D&VA.Q+LB.PC 815 #			
CLR.SD&SS 1560 \$ CONSOLE.MODE? 1647 \$ D&Q_D+Q 806 \$ D&RCCI_PC 807 \$ D&VA_ALU 808 \$ D&VA_D+LC 809 \$ D&VA_D+C 810 \$ D&VA_D+C 811 \$ D&VA_LA 811 \$ D&VA_LA 812 \$ D&VA_LA 812 \$ D&VA_LA 813 \$ D&VA_LB 813 \$ D&VA_C 814 \$ D&VA_C 815 \$ D&VA_C 815 \$ D&VA_C 816 \$ D&VA_C 817 \$ D&VA_C 817 \$ D&VA_C 818 \$ D			
CONSOLE.MODE? D&QD+Q D&RCCI_PC BO7 # D&VAALU BO8 # D&VAD+LC D&VAD+Q D&VAD+C D&VALA D&VALA B11 # D&VALB B2VALB B2VAQ B13 # D&VAQ B14 # D&VAQ B15 # D&VAQ B15 # D.(1)? D.BO?			
D2Q_D+Q			
D&RCCIPC 807 # D&VA_ALU 808 # D&VA_D+LC 809 # D&VA_D+Q 810 # D&VA_D+KCI 811 # D&VA_LA 812 # D&VA_LB 813 # D&VA_Q+LB 814 # D&VA_Q+LB 815 # D(1)? 1649 # D.80? 1650 #			
D2VAALU 808			
D&VAD+LC			
D&VA_D+Q 810 # D&VA_D-KC] 811 # D&VA_LA 812 # D&VA_LB 813 # D&VA_Q 814 # D&VA_Q+LB.PC 815 # D(1)? 1649 # D.BO? 1650 #			
D2VA_D-KC] 811 # D8VA_LA 812 # D8VA_LB 813 # D8VA_Q 814 # D8VA_Q+LB.PC 815 # D(1)? 1649 # D.BO? 1650 #			
D&VALA 812 # D&VALB 813 # D&VAQ 814 # D&VAQ 815 # D&VAQ 1649 # D.BO? 1650 #			
D&VA_LB 813 # D&VA_Q 814 # D&VA_Q+LB.PC 815 # D(1)? 1649 # D.BO? 1650 #			
D&VA_Q D&VA_Q+LR.PC 815 # D(1)? 1649 # D.BO? 1650 #			
D&VA_Q+LB.PC 815 # D(1)? 1649 # D.BO? 1650 #			
D(1)? 1649 # D.BO? 1650 #			
D.BO? 1650 #			
D.B1? 1651 *			
	D.B17	1651 #	

# NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06
9	Cross Reference Listing - Macro Names
D.B2?	1652 #
D.BYTES?	1653 #
D.NE.0?	1654 #
DOT	1655 #
D2-0?	1656 # 1657 #
D27 D3-07	1658 #
D31?	1659 #
1137	1660 #
DATA.TYPE?	1661 #
DBL?	1662 #
DEJ_CACHE	817 # 1794
DEJ_CACHE.IBCHK	818 #
DCJ_CACHE.LK	819 #
DED_CACHE.NOCHK	820 #
DEJ_CACHE.P	821 #
DC3_CACHE.WCHK	822 *
D_0	824 #
DO+K[3+1	825 #
D_O+LC+1	826 #
D_0-D	827 #
D-0-KC3	828 #
DO-Q	829 * 830 *
D_O-Q-1 D_ACCEL%SYNC	831 #
D_ALU	832 #
D_ALU(FRAC)	833 #
D_ALU.LEFT	834 #
D_ALU.LEFT2	835 #
D_ALU.LEFT3	836 #
D_ALU.RIGHT	837 #
D_ALU.RIGHT2	838 #
D_BLANK	839 #
D_CACHE.INST.DEP	840 #
D_CACHE.LKC3	841 #
D_CACHE.WCHKEJ	842 #
D_CACHECI	843 #
D_D(FRAC) D_D+KC3	844 # 845 #
D_D+KE3+1	846 #
D_D+LB	847 #
D_D+LC	848 #
D_D+LC+PSL • C	849 #
D_D+Q	850 #
D_D+Q+1	851 #
D_D-KC3	852 #
D_D-LC	853 #
D_D-Q	854 +
D_D-Q-1	855 #
D_D.OXTC3	856 #
D_D.OXTE3+KE3 D_D.OXTE3+Q	857 * 858 *
D_D.OXTEJ+0 D_D.OXTEJ+0+1	859 #
D_D.OXTEG.ANDNOT.KEG	860 #
D_D.OXTEJ.OR.Q	861 #

Fase 48

	VEDEOD 41 (AD) 40 IAN 00 4/14E1A/
, NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06 Cross Reference Listing - Macro Names
•	CLOS2 Keletelice Firming - uecto Hemes
D_D.OXTE3.XOR.Q	862 #
D_D.OXTEJ.XOR.RCEJ	863 #
D_D.AND.KE3	864 #
D_D.AND.KCJ.LEFT2	865 #
D_D.AND.KCJ.RIGHT	866 #
D_D.AND.LC	867 #
D_D.AND.MASK	868 #
D_D.AND.Q	869 #
D_D.AND.RCE3	870 #
D_D.ANDNOT.KC3	871 #
D_D.ANDNOT.LC	872 #
D_D.ANDNOT.PSWZ	873 #
D_D.ANDNOT.Q	874 #
D_D.ANDNOT.RCEJ	875 #
D_D.LEFT	876 #
D_D.LEFT2	877 #
D_D.OR.ASCII	878 #
D_D.OR.KCJ	879 #
D_D.OR.PSWC	880 * 881 *
D_D.OR.PSWV	882 *
D_D.OR.Q D_D.OR.RCEJ	883 #
D_D.OR.RCJ	884 #
D_D.ORNOT.MASK	885 #
D_D.RIGHT	886 #
D_D.RIGHT(B)	887 #
D_D.RIGHT2	888 #
D_D.SWAP	889 #
D_D.SXTCJ	890 #
D_D.SXTCJ.RIGHT	891 #
D_D.XOR.KE3	892 #
D_D.XOR.LC	893 #
D_D.XOR.Q	894 #
D_DAL.NORM	895 #
D_DAL.SC	896 #
D_DEGKEG	897 \$
D_DCJMASK	898 #
D_DEJQ D_INT.SUM	899 * 900 *
D_K[]	901 #
D_KE3.RIGHT	902 #
D_KCJ.RIGHT2	903 #
D_LA	904 #
D_LA(FRAC)	905 #
D_LA+D+PSL.C	906 #
D_LA-D	907 #
D_LA-KCJ	908 #
D_LA.AND.KE3	909 #
D_LA.RIGHT	910 #
D_LB	911 #
D_LB.FC	912 #
D_LC	913 #
D_LC(FRAC)	914 #
D_NOT.D	915 #
D_NOT.KE3	916 #

; NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06	Pase	49
\$	Cross Reference Listins - Macro Names		
T NOT MACK	917 🛊		
D_NOT · MASK			
D_NOT • Q	918 #		
D_NOT.RE3	919 #		
D_PACK.FP	920 #		
D_PACK.FP.LEFT	921 #		
D_PC	922 #		
D_PC.LEFT	923 🛊		
D_Q	924 #		
D_Q(FRAC)	925 🛊		
D_Q+D	926 #		
D_Q+KC3	927 #		
D_Q+LB	928 #		
D_Q+PC	929 #		
n_q-p	930 #		
DQ-D-1	931 *		
D_Q-KCJ	932 *		
D_Q-KCJ-1	933 *		
D_Q-PCSV	934 *		
n_g.oxtc3	935 *		
D_Q.AND.KCJ	936 #		
D_Q.AND.LC	937 #		
D_G.AND.MASK	938 #		
D_Q.AND.RCEJ	939 #		
D_G.ANDNOT.D	940 #		
D_Q.ANDNOT.KCJ	941 #		
	942 #		
D.G.ANDNOT.MASK	943 #		
D_G.ANDNOT.PSWC			
D.G.ANDNOT.PSWN	944 #		
D_Q.ANDNOT.PSWZ	945 #		
D_Q.LEFT	946 #		
D_Q.OR.KCJ	947 #		
D_Q.OR.PSWC	948 #		
D_Q.OR.RCE3	949 #		
D_Q.ORNOT.MASK	950 #		
D_G.RIGHT	951 #		
D_Q.RIGHT2	952 #		
D_Q.SXTCJ	953 #		
D_Q.XOR.RCC3	954 #		
D_0C3D	955 #		
D_QCJKCJ	956 #		
D_QCJMASK	957 #		
D_R(PRN+1)	958 🛊		
D_R(SC)	959 #		
D_R(SP1+1)	960 #		
D_RC(SC)	961 #		
DERCED	962 #		
D_RLOG	963 #		
D_RLOG.RIGHT	964 #		
D_RC3	965 # 1767		
D_REJ(FRAC)	966 🛊		
D_RCJ.AND.KCJ	967 #		
D_RCJ.OR.KCJ	968 #		
D_REJ.ORNOT.KEJ	969 #		
E.FORK	1562 #		
EALU.N?	1664 *		

; NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06 Cross Reference Listing - Macro Names	Page	50
EALU.Z?	1665 #		
EALU?	1666 #		
EALU_D(EXP)	971 *		
EALU_FE	972 #		
EALU_KED	973 #		
EALU_REJ(EXP)	974 #		
EALU_SC	975 #		
EALU_SC+FE	976 #		
EALU_SC+KEJ	977 #		
EALU_SC-FE	978 #		
EALUSC-K[]	979 🛊		
EALU_SC.ANDNOT.KEJ	980 #		
EALU_STATE	981 #		
END.DP1?	1667 ≢		
EXCEPT.ACK	1563 ≢		
FE&SC_KEJ	983 #		
FE_0(A)	984 #		
FE_D(EXP)	985 #		
FELEALU	986 🛊		
FE_KC1	987 🛊		
FE_LA(EXP)	988 #		
FE_NARS(SC-FE)	989 #		
FE_NABS(SC-LA(EXP))	990 #		
FE_Q(EXF)	991 #		
FE_RCJ(EXP)	992 *		
FESC	993 #		
FE_SC+1	994 #		
FE_SC+FE	995 #		
FE_SC+KCJ	996 #		
FE_SC+LA(EXP)	997 #		
FE_SC-FE	998 #		
FE_SC-KCJ	999 #		
FE_SC-LA(EXP)	1000 #		
FE_SC-SHF.VAL	1001 * 1002 *		
FE_SC.ANDNOT.FE	1003 #		
FE_SC.ANDNOT.KCJ	1004 #		
FELSC.OR.KED	1005 ‡		
FE_SHF.VAL FE_STATE	1005 ‡		
FLUSH.IB	1565 ‡		
FPD?	1667 #		
G.FORK	1567 #		
IB.TEST?	1671 #		
ID(SC)_D	1008 #		
TDC 3_D	1009 #		
ID_D&NO.SYNC	1010 #		
ID_D.SYNC	1011 #		
INHIBIT.IB	1569 #		
INT?	1672 # 1784		
INTERRUPT.REQ?	1673 #		
INTRPT.ACK	1570 #		
INTRPT.STROBE	1571 # 1776 1837		
IRO.C31?	1674 #		
IRO?	1675 #		
IR1?	1676 #		
	\cdot		

; NEWSAM.MCR	MICRO2 1L Cross Refe			82 16:15:06 Macro Names			Pase	51
IR2-17	1677 #							
TRD	1572 #							
TRD.11	1573 #							
IRDO	1574 #							
IRD1	1575 #							
KCI	1013 #							
LAB_R(DST)	1015							
LAB_R(PRN)	1016							
LAB_R(PRN+1)	1017 #							
LAB_R(SC)	1018 #							
LAB_R(SP1)	1019 #							
LAB_R(SP1+1)	1020 #							
LAB_R1&RCCJO	1021 #							
LAB.R1&RCCJ_O+LC+1	1022 #							
LAB_R1%RCC3_O-D	1023 #				*			
LAB_R1&RCCJ_ALU	1024 #	1771						
LAB_R1&RCCJ_ALU.RIGHT2	1025 #							
LAB_R1&RCCJ_D+LC	1026 #							
LAB_R1&RCEJ_D.OXTEJ+KEJ	1027 #							
LAB_R1&RCCJ_Q-KCJ	1028 #							
LAB_RCJ	1029 #							
LAST.REF?	1679 #							
	1031 #							
LA_R(DST)&LB_R(SRC)								
LA_R(SP2)%LB_R(SP1)	1032 #							
LA_RACJ	1033 #	1808						
LC_RC(SC)	1034 #							
LCLRCCC	1035 #	1783						
LC_RCC3&R1_(LA+LB).LEFT	1036 #							
LC_RCE3&R1_(LA+LB+PSL+C)+LEF	T 1037 #							
LC_RC[]&R1_(LA+LB.RLOG).LEFT	1038 #							
LC_RCEJ&R1_(LA-LB).LEFT	1039 #							
LC_RCCJ&R1_(LA-LB.RLOG).LEFT	1040 #							
LC_RCC3&R1_ALU	1041 #							
LC_RCEJ&R1_D	1042 #							
LC_RCCJ&R1_LA+KCJ	1043 #							
LC_RCEJ&R1_LA-KEJ	1044 #							
LC_RCEJ&R1_LB	1045 #							
LC_RCC1&R1_Q	1046 #							
LDAD.ACC.CC	1577 #							
LOAD.IB	1578 #							
LOAD.IB.11	1579 #							
LONG	1580 #	1788	1807	1831				
MEMORY.NOP	1582 #	1700	100/	# 43 th T				
MODE.LSS.ASTLVL?	1681 #							
MUL.OXT	1583 #							
MUL.1XT	1584 #							
MUL?	1682 #							
MULM DONE	1585 #							
MULP . DONE	1586 #							
N&Z_ALU	1048 #							
N&Z_ALU.V&C_O	1049 #							
NEST.ERR?	1684 #							
N_AMX.Z_TST	1050 #				*			
PC&VA_ALU	1052 #							
PC&VA_D	1053 #							

; NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06	Page	52
•	Cross Reference Listins - Macro Names		
•			
PC&VA_D+KE3	1054 #		
FC&VA_D-KE3	1055 #		
PC&VA_D-PC	1056 #		
PC&VA_D.OXTE3	1057 #		
PC&VA_D.OXTEJ+PC	1058 #		
PC&VA_D.SXTEJ+PC	1059 #		
PC&VA_KEJ	1060 #		
PC&VA_PC	1061 #		
PC&VA_Q	1062 #		
PC&VA_Q+PC	1063 #		
PC&VA_Q-D	1064 #		
PC&VA_Q-KEJ	1045 #		
PC&VA_Q.SXTCJ+PC	1066 *		
PC&VA_RCE3	1067 #		
PC&VA_REJ.ANDNOT.KEJ	1068 #		
PC.MODES?	1686 #		
PC_PC+1	1070 # 1789 1832		
PC_PC+2	1071 #		
PC_PC+4	1072 #		
PC_PC+N	1073 #		
PC_Q+PC	1074 #		
PC_VA	1075 #		
PC_VIRA	1076 #		
POLY DONE	1598 #		
PSL.C?	1687 #		
PSL.CC?	1688 #		
PSL.MODE?	1689 \$		
PSL · N?	1690 #		
PSL.V?	1691 #		
PSL.Z?	1692 #		
PSL <c>_AMX0</c>	1077 #		
PTE.VALID?	1693 #		
(28VA_ALU	1079 #		
Q8VA_D	1080 #		
Q&VA_D+LC	1081 #		
Q&VA_LA	1082 #		
Q&VAQ+LB+PC	1083 #		
Q31?	1695 #		
QD_(Q+LB)D.RIGHT2	1085 #		
QD_(Q+LC)D.RIGHT2	1086 #		
QD_(Q-LB)D.RIGHT2	1087 #		
QD_(Q-LC)D.RIGHT2	1088 #		
QD_QD.RIGHT2	1089 #		
QUAD?	1696 #		
Q_(LA+Q).RIGHT	1091 # 1775		
Q(Q+LB).RIGHT	1092 # 1836		
0_0	1093 #		
Q_O+LC+1	1094 #		
Q_0+MASK+1	1095 #		
Q_O+PC.RLOG	1096 #		
Q_O-D	1097 #		
G_O-KEJ	1098 *		
G-0-r.c	1099 #		
0-0-0	1100 #		
Q_ACCEL &SYNC	1101 #		

Pase 53

• NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06
9	Cross Reference Listins - Macro Names
Ch. Al. II	1102 #
Q_ALU	1103 #
Q_ALU(FRAC)	1103 #
Q_ALU.LEFT	1105 #
Q_ALU.LEFT2	1106 #
QLALU.LEFT3 QLALU.RIGHT	1107 #
Q_ALU.RIGHT2	1108 #
G_D	1109 #
Q_D(FRAC)(B)	1110 #
Q_D+KEJ	1111 #
Q_D+KEJ+1	1112 #
Q_D+KEJ.LEFT	1113 #
Q_D+LC	1114 #
Q_D-KCJ	1115 #
Q_D-LC	1116 #
Q_p-q	1117 #
CJTXO.Q_D	1118 4
Q_D.OXTEJ+KEJ.LEFT	1119 #
Q_D.OXTEJ.OR.PACK.FP	1120 #
G_D.AND.KEJ	1121 #
Q_D.AND.KEJ.RIGHT	1122 #
Q_D.AND.KCJ.RIGHT2	1123 #
Q_D.AND.RCE3	1124 #
Q_D.ANDNOT.RCEJ	1125 #
Q_D.LEFT3	1126 #
Q_D.OR.KE3	1127 #
Q_D.OR.RCE3	1128 #
Q_D.RIGHT	1129 #
Q_D.RIGHT2	1130 #
Q_D.SXTCJ	1131 +
Q_D.XOR.Q	1132 #
Q_DEC.CON_	1133 #
Q_IB.BDEST	1134 * 1135 *
Q_IB.DATA	1136 #
Q_ID(SC) Q_IDE3	1137 *
G_KE3	1138 #
Q_KE3+1	1139 #
Q_KEJ.CTX	1140 #
Q_KCJ.RIGHT	1141 #
Q_KIJ.RIGHT2	1142 #
QLLA	1143 #
Q.LA+KEJ	1144 #
QLA+Q	1145 #
QLA-KEJ	1146 #
Q_LA.AND.KEJ	1147 #
QLLA.ANDNOT.RCC3	1148 #
Q_LB	1149 #
Q_LC	1150 #
Q_NOT+Q	1151 #
QNOT.RED	1152 #
Q_PACK.FP	1153 #
QLPC	1154 #
Q_Q(FRAC)	1155 #
Q_Q(FRAC)(B)	1156 #

F NEWSAM.MCR	MICRO2 1L(O2) 18-JAN-82 16:15:06 Cross Reference Listing - Macro Names
Q_Q+D	1157 #
Q_Q+KC3	1158 # 1820
Q_Q+K[]+1	1159 #
Q_Q+LC	1160 #
G_Q+PC	1161 #
Q_Q-D	1162 #
Q_Q-D-1	1163 #
Q_Q-KC3	1164 # 1825
Q_Q-K[3-1	1165 #
Q_Q-LC	1166 #
G_G-LC-1	1167 #
Q_Q-MASK-1	1168 #
Q_Q.OXTCJ-KCJ	1169 #
Q_Q.OXTCJ.LEFT	1170 #
Q_Q.OXTCJ.OR.D	1171 #
Q_Q.AND.KCJ	1172 #
Q_Q.AND.KCJ.RIGHT	1174 #
Q_Q.AND.KCJ.RIGHT2	1173 #
Q_Q.AND.RCE3	1176 #
Q_Q.AND.RE3	1175 #
Q_Q.ANDNOT.D	1177 #
Q_Q.ANDNOT.KCJ	1178 # 1781
Q_Q.ANDNOT.RCE3	1179 #
Q_Q.LEFT	1180 #
Q_Q.LEFT2	1181 #
Q_Q.OR.KE3	1182 #
Q_Q.ORNOT.MASK	1183 #
Q_Q.RIGHT	1184 #
Q_Q.RIGHT2	1185 #
Q_Q.SXTC3	1186 #
Q_Q.XOR.KE3	1187 #
Q_R(FRN).ANDNOT.Q	1188 #
Q_R(PRN+1)	1189 #
Q_R(PRN+1).AND.Q	1190 #
G_R(SC)	1191 * 1192 *
Q_R(SRC!1).AND.KE3 Q_RC(SC)	1193 *
Q_RCE3	1194 #
Q_RCEJ(FRAC)	1195 #
Q_REJ	1196 \$ 1763
Q_REJ(FRAC)	1197 #
Q_REJ.AND.KEJ	1198 ‡
Q_REJ.AND.KEJ.RIGHT	1199 #
Q_RCJ.ANDNOT.KCJ	1200 \$
Q_RCJ.OR.KCJ	1201 #
Q_SC	1202 \$
Q_SHF	1203 #
R(DST)_ALU	1205 #
R(DST)_D	1206 #
R(DST)_D.SXTCJ.RIGHT	1207 #
R(PRN)_O+D.RLOG	1209 #
R(PRN)_ALU	1210 \$
R(PRN)_D	1211 #
R(PRN)_D+KEJ.RLOG	1212 #
R(PRN)_D-KEJ.RLOG	1213 #

Pase 55

; NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06 Cross Reference Listins - Macro Names
R(PRN)_D.OR.Q	1214 #
R(PRN)_DEJQ	1215 #
R(PRN)_KEJ	1216 #
R(PRN)_LA+KCJ.RLOG	1217 #
R(PRN)_LA+Q	1218 #
R(PRN)_LA-KEJ+RLOG	1219 #
R(PRN)_LACIMASK	1220 #
R(PRN)_LC	1221 #
R(PRN)_PACK.FP	1222 *
R(PRN)_Q R(PRN)_Q+KCJ.RLOG	1223 + 1224 +
R(PRN)_Q-K[].RLOG	1225 #
R(PRN+1)_ALU	1226 #
R(PRN+1)_D	1227 #
R(PRN+1)_D.OR.Q	1228 #
R(PRN+1)_KED	1229 #
R(PRN+1)_LA	1230 #
R(PRN+1)_LC	1231 #
R(PRN+1)_Q	1232 #
R(SC)_ALU R(SC)_D	1234 * 1235 *
R(SC)_KEJ	1236 #
R(SC)_LA	1237 #
R(SC)_LA+D	1238 #
R(SC)_LA-D	1239 #
R(SC)_LC	1240 #
R(SC)_Q	1241 #
R(SP1)ALU	1243 #
R(SP1)_D	1244 #
R(SP1)_KC] R(SP1)_PACK.FP	1245 # 1246 #
R(SP1)Q	1247 #
R(SP1+1)_LC	1248 #
R(SP1+1)_Q	1249 #
R(SRC!1)_ALU	1251 #
R(SRC!1)_D(B)	1252 #
R(SRC)_ALU	1253 #
R(SRC)_D	1254 #
R(SRC)_D(B)	1255 #
R(SRC)_D+KCJ.RLOG R(SRC)_D-KCJ.RLOG	1256 # 1257 #
R(SRC)_LC	1258 #
R(SRC)_Q	1259 #
R6_D+KCJ.RLOG	1261 #
R6_LA+KEJ+RLOG	1262 #
R6_LA-KEJ.RLOG	1263 #
RC(SC)_O-LC	1265 #
RC(SC)_ALU	1266 #
RC(SC)_ALU+RIGHT RC(SC)_D	1267 # 1268 #
RC(SC)D RC(SC)Q	1268 #
RCC38VA_D+Q	1271 #
RCEJ_O	1272 #
RCCJ_O+KCJ+1	1273 #
RCE3_0+LC+1	1274 #

F NEWSAM.MCR	MICRO2 1L(O2) 18-JAN-82 16:15:06 Cross Reference Listins - Macro Names
RCEJ_O+MASK+1	1275 #
RCEJ_O+MASK+1.RIGHT2	1276 #
RCEJ_O-D	1277 #
RCCJ_ALU	1278 #
RCCJ_ALU.LEFT	1279 #
RCCJ_ALU.LEFT2	1280 #
RCCJ_ALU.LEFT3	1281 #
RCCJ_ALU.RIGHT	1282 #
RCEJ_ALU.RIGHT2	1283 #
RCCJ_D	1284 * 1285 *
RCC1_D(B)	1286 #
RCC1_D+KC1	1287 #
RCCI_D-KCI RCCI_D.OXTCI	1288 #
RCE3_D.AND.KE3	1289 #
RCEJ_D.AND.MASK	1290 #
RCC1_D.ANDNOT.Q	1291 #
RCIJ_D.CTX	1292 #
RCCJ_D.LEFT	1293 #
RCCJ_D.LEFT3	1294 #
RCCD_D.OR.KCD	1295 #
RCC1_D.OR.Q	1296 #
RCCJ_D.ORNOT.KCJ	1297 #
RCC1_D.SXTC1	1298 #
RCCI_KCI	1299 #
RCC1_KC1+1	1300 + 1301 +
RCCJ_KCJ.LEFT2	1302 #
RCC1_KC1.LEFT3	1302 *
RCCJ_KCJ.RIGHT2 RCCJ_LA	1304 #
RCC J_LA+LB+CTX	1305 #
RCEJ_LA-KEJ	1306 #
RCEI_LA.AND.KEI	1307 #
RCCJ_LA.CTX	1308 #
RCCILLB	1309 #
RCC1_LB.LEFT	1310 #
RCCD_LC	1311 #
RCE3_NOT.Q	1312 #
RCIJ_PACK+FP	1313 + 1314 +
RCCJ_PC RCCJ_Q	1315 #
RC[]_Q+1	1316 #
RCCI_Q+KCI	1317 #
RCC 1_0+LC	1318 #
RCCJ_Q+PC	1319 #
RCE1Q+FC+1	1320 #
RCC 1Q-KC I	1321 #
RCCJ_Q-LC	1322 #
RCCJ_Q-MASK-1	1323 #
RCEJ_G.OXTEJ	1324 #
RCCJ_G.AND.KCJ	1325 #
RCEJ_Q.ANDNOT.KEJ	1326 #
RCC1_0.LEFT	1327 #
RCC1_Q.LEFT3 RCC1_Q.RIGHT	1328 * 1329 *
IVAL TERMINATION	4 W M / M

P NEWSAM.MCR	MICRO2 1L(O2) 18-JAN-82 16:15:06 Cross Reference Listins - Macro Names
RCEJ_Q.RIGHT2	1330 #
RCIJ_Q.SXTIJ	1331 *
RCEL_RLOG.RIGHT	1332 #
RETURNO	1590 #
RETURN1	1591 #
RETURN10	1592 #
RETURN100	1593 #
RETURN10C	1594 #
RETURN10E	1595 #
RETURN12	1596 #
RETURN18	1597 #
RETURN1F	1598 * 1599 *
RETURN2 RETURN20	1600 #
RETURN24	1601 #
RETURN3	1602 #
RETURN4	1603 #
RETURN40	1604 #
RETURN60	1605 #
RETURN61	1606 #
RETURN8	1607 #
RETURN9	1608 #
RETURNF	1609 #
RETURNOS	1610 #
RLOG.EMPTY?	1698 #
ROR?	1699 #
REJ&VA_LA+KEJ	1334 #
REJ&VALLA-KEJ REJ&VALLA-KEJ+RLOG	1335 * 1336 *
RED&VALEN-RED*REGG	1337 #
RC3_0	1338 #
RC3_0+LB+1	1339 #
REJ_0-1	1340 #
RED_O-D	1341 #
RCJ_O-KCJ	1342 #
RCJ_O-LB	1343 #
RE30-Q	1344 #
REJ_ALU	1345 # 1821 1826
RED_ALU.LEFT RED_ALU.LEFT3	1346 # 1347 #
REJ_ALU.RIGHT	1348 #
REJ_ALU.RIGHT2	1349 #
RCJ_D	1350 #
RCJ_D+KCJ	1351 #
RCJ_D+Q	1352 #
RC3_D+Q+1	1353 #
RED_D-KCD	1354 #
RC3_D-LC-1	1355 #
RCJ_D-Q	1356 #
RCJ_D.AND.KCJ RCJ_D.OR.LC	1357 # 1358 #
REJ_D.OR.PACK.FP	1359 #
REJ_D.OR.Q	1360 *
REJ_KEJ	1361 *
REJ_LA ·	1362 #
7 2 mg 40 100 mm 7 7	

J NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06	Page	58
\$	Cross Reference Listins - Macro Names		
REJ_LA+D	1363 #		
RED_LA+D+1	1364 #		
RCJ_LA+KCJ	1365 ‡		
RCJ_LA+KCJ+1	1366 ‡		
REJ_LA+KEJ.RLOG	1367 #		
RED_LA+LC	1368 #		
RCJ_LA+MASK+1	1369 #		
REJ_LA+Q	1370 #		
REJ_LA-D	1371 #		
REJ_LA-KEJ	1372 #		
REJ_LA-KEJ.RLOG	1373 #		
REJ_LA-MASK-1	1374 #		
REJ_LA-Q	1375 #		
REJ_LA.AND.KEJ	1376 #		
RED_LA.OR.D	1377 #		
REJ_LA.ORNOT.MASK	1378 #		
REJULB	1379 #		
REJLLC	1380 #		
REJLLC.RIGHT	1381 #		
RED_NOT.0	1382 #		
RED_NOT+D	1383 #		
RED_NOT.MASK	1384 #		
REILNOT • Q	1385 #		
RED_PACK.FF	1386 #		
REDLO	1387 # 1830		
RC3_Q+1	1388 #		
RC3_Q+5	1389 #		
RCJ_Q+KCJ	1390 #		
REJ_Q+LB	1391 *		
REJ_Q+LC	1392 #		
RC1_Q-D	1393 #		
RCJ_Q-D-1	1394 * 1395 *		
REILQ-KEI	1396 #		
REJ_Q-KEJ.RLOG REJ_Q-LC	1397 ‡		
REJ_Q.AND.KEJ	1398 #		
REJ_Q.ANDNOT.KEJ	1399 #		
RCJ_Q.OR.D	1400 #		
REJ_Q.ORNOT.KEJ	1401 \$		
REJ_Q.RIGHT.1	1402 #		
REI_RLOG.RIGHT.1	1403 #		
SC&STATE_STATE-RCJ(EXF)	1405 #		
SC.GT.O?	1701 #		
SC.NE.O?	1702 #		
SC?	1703 #		
SC_O(A)	1406 #		
SCO-KEI	1407 #		
SCALU	1408 #		
SCLALU(EXF)	1409 #		
SC_D	1410 #		
SCLD(EXP)	1411 #		
SC_D(EXP)(A)	1412 #		
SC_D(EXP)(B)	1413 #		
SCLD-KEI	1414 #		
SC_D.OXTED-KED	1415 #		

; NEWSAM.MCR	MICRO2 1L(O2) Cross Reference	18-JAN-82 16:15:06 Listing - Macro Names
SC_D.OXTEJ.XOR.KEJ	1416 #	
SC_D.AND.KED	1417 #	4
SC_D.OR.KCJ	1418 #	
SC_D.SXTCI	1419 #	
SC_EALU	1420 #	
SCLEE	1421 # 1422 #	
SCLKEJ SCLKEJ-ALU	1423 #	
SCLLA	1424 #	
SC_LA.AND.KE3	1425 #	
SC_LC(EXP)	1426 #	
SC_NABS(SC-FE)	1427 #	
SC_FSLADDR	1428 #	
SC_Q	1429 #	
SC_Q(EXP)	1430 #	
SC_Q(EXF)(B)	1431 #	
SC_Q+KC]	1432 # 1433 #	
SC_Q-KEJ	1434 #	
SC_Q.AND.KE3 SC_Q.OR.KE3	1435 #	
SC_Q.SXTCJ	1436 #	
SC_RCEI	1437 #	
SC_RCEI(EXP)	1438 #	
SCLRED	1439 #	
SC_REJ(EXP)	1440 #	
SCLREJ:AND:KEJ	1441 #	
SC_SC+1	1442 #	
SC_SC+EXF(Q)(A)	1443 #	
SC_SC+FE	1444 # 1445 #	
SC_SC+KEJ SC_SC+SHF•VAL	1446 #	
SC_SC-FE	1447 #	
SCLSC-KED	1448 #	
SC_SC-SHF.VAL	1449 #	
SC_SC.ANDNOT.FE	1450 #	
SC_SC.ANDNOT.KEJ	1451 #	
SC_SC.OR.KEI	1452 #	
SC_SHF.VAL	1453 # 1454 #	
SC_STATE SC_STATE.ANDNOT.KEJ	1455 #	
SC_STATE.OR.KCJ	1456 #	
SD_NOT.SD	1457 #	
SD_SS	1458 #	
SET.CC(BYTE)	1612 #	
SET.CC(INST)	1613 #	
SET.CC(LONG)	1614 #	
SET.CC(ROR)	1615 #	•
SET.CC(WORD)	1616 #	
SET.FPD SET.NEST.ERR	1617 # 1618 #	
SET.PSL.C(AMX)	1619 #	
SET.V	1620 #	
SIGNS?	1704 #	
SPEC	1621 #	
SPECG	1622 #	

F NEWSAM.MCR		.(02) Prence	18-JAN-82 16:15:06 Listing - Macro Names
400 508	1705 #		
SRC.FC?	1705 #		
SS? SS_0&SD_0	1459 #		
95_0451_0 95_ALU15	1460 #		
SS_SD	1461 #		
SS_SS.XOR.ALU15&SD_ALU15	1462 #		
START. IB	1623 #		
STATE(7)?	1707 #		
STATEO?	1708 #	1777	1838
STATE1-0?	1709 #		
STATE1?	1710 #		
STATE2?	1711 #		
STATE3-07	1712 # 1713 #		
STATE3? STATE4?	1714 #		
STATES?	1715		
STATE6?	1716 #		
STATE7-4?	1717 #		
STATE_O(A)	1463 #		
STATE_AMX.EXP	1464 #		
STATE_D(EXF)	1465 #		
STATELFE	1466 #		
STATE_FIRST	1467 #		
STATEINNEROBJ	1468 #		
STATE_INNERSRC	1469 #		
STATE_KCJ	1470 #	1764	1815
STATE_OUTER STATE_PREDEC	1471 # 1472 #		
STATE_Q(EXP)	1473 #		
STATE_SC.VIA.KMX	1474 #		
STATE_SKPLONG	1475 #		
STATE_STATE+1	1476 #		
STATE_STATE+FE	1477 #		
STATE_STATE+KCJ	1478 #		
STATE_STATE-FE	1479 #		
STATE_STATE-KCJ	1480 #		
STATE_STATE.AN.5TOO	1482 #		
STATE_STATE.AN.6TO4 STATE_STATE.AN.DESTDBL	1483 # 1484 #		
STATE_STATE.AN.NOTPREDEC	1485 #		
STATE_STATE.AN.PREDECZERO	1486 #		
STATE_STATE.AN.SKPLONG	1481 #		
STATE_STATE.ANDNOT.FE	1487 #		
STATE_STATE.ANDNOT.KEJ	1488 #		
STATE_STATE.ANDNOT.SHF.VAL	1489 #		
STATE_STATE.OR.ADJINP	1492 #		
STATE_STATE.OR.DEST	1493 #		
STATE_STATE.OR.DESTDBL	1494 #		
STATE_STATE.OR.FE STATE_STATE.OR.FILL	1490 #		
STATE_STATE.OR.FILL STATE_STATE.OR.FLOAT	1495 # 1496 #		
STATE_STATE.OR.FLUAT	1496 #		
STATE_STATE.OR.MOVE	1497 #		
STATE_STATE.OR.FATT1	1498 #		
STATE_STATE.OR.PATT2	1499 #		

Pase 61

STOP.IB	9 NEWSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:15:06 Cross Reference Listins - Macro Names
SMAPD	STOP.IB	1624 #
TB.TEST? 1719 # TEST.TB.RCHK 1626 # TEST.TB.RCHK 1627 # TRAP.ACCL] 1628 # UA31-307 1721 # UA31-307 1722 # UA31-307 1722 # UA_ALU 1502 # 1782 UA_D 1503 # UA_D 1503 # UA_D 1503 # UA_D+KC] 1504 # UA_D+C 1505 # UA_D+C 1506 # UA_D-ANDNOT.KC] 1508 # UA_LA+C] 1509 # UA_LA+C] 1509 # UA_LA+C] 1509 # UA_LA+C] 1510 # UA_LA+C] 1511 # UA_LA+KC] 1512 # UA_LA+C] 1513 # UA_LA+C] 1514 # UA_LA-D 1515 # UA_LA-D 1516 # UA_LA-C] 1517 # UA_LA-C] 1519 # UA_LA-C] 1519 # UA_LA-RC] 1519 # UA_LA-RC] 1519 # UA_LA-RDNNOT.KC] 1522 # UA_LA-RDNNOT.KC] 1521 # UA_LA-RDNNOT.KC] 1522 # UA_LA-RDNNOT.KC] 1524 # UA_LA-RDNNOT.KC] 1525 # UA_CA-C 1520 # U		
TEST.TB.RCHK		
TEST.TB.WCHK TRAP.ACCTI TRAP.ACCTI TRAP.ACCTI 1628		
TRAP.ACCE] 1628		1627 #
VA31-30? 1721 # 1722 # 1782		1628 #
VA_ALU		1721 #
VA_D VA_D+KCJ VA_D+LC 1504	VA317	1722 #
VA_D+KCJ	VA_ALU	1502 # 1782
VA_D+LC VA_D+Q VA_D-OXTEJ+Q VA_D-ANDNOT-KEJ VA_D-ANDNOT-KEJ VA_EA VA_EB VA_EA VA_EB VA_EA VA_EB VA_EA VA_EB	VALD	1503 #
VA_D+Q VA_D, OXTC]+Q VA_D, OANDOT, KC] VA_CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC	VA_D+KE3	
VA_D.ANDNOT.KC] 1508 # VA_D.ANDNOT.KC] 1508 # VA_KC] 1509 # VA_LA VA_LA VA_LA VA_LA+D 1511 # VA_LA+KC] 1512 # VA_LA+KC] 1513 # VA_LA+C] 1513 # VA_LA+C] 1515 # VA_LA+C] 1516 # VA_LA+C] 1516 # VA_LA+C] 1516 # VA_LA-C] 1517 # VA_LA-C] 1519 # VA_CA-C] 1523 # VA_CA-C 1520 # VA_CA-C 152	VA_D+LC	
VA_NCI	VAD+Q	
VA_KEJ	VA_D.OXTEJ+Q	
VA_LA+D VA_LA+RCJ+1 VA_LA+KCJ+1 VA_LA+CD VA_LA+CD VA_LA+CD VA_LA+CD VA_LA-CD VA_CD	VA_D.ANDNOT.KCJ	
VA_LA+KC]	VAKCJ	
VA_LA+KC]		
VA_LA+KCJ+1 VA_LA+C VA_LA+C VA_LA-D VA_LA-C		
VA_LA+PC		
VA_LA+U VA_LA-D VA_LA-KI] VA_LA-KI] VA_LA-KI]-1 VA_LA-KI]-1 VA_LA-Q 1519		
VA_LA-KI] 1516 # VA_LA-KI]-1 1518 # VA_LA-KI]-1 1518 # VA_LA-AND.LC 1520 # VA_LA.AND.LC 1520 # VA_LA.ANDNOT.KI] 1521 # VA_LB+D.OXT 1522 # VA_C 1523 # VA_C 1523 # VA_C 1525 # VA_C 1525 # VA_C 1526 # VA_C 1527 # VA_C 1528 # VA_C 1528 # VA_C 1529 # VA_C 1530 # VA_C 1531 # VA_C 1531 # VA_C 1531 # VA_C 1532 # VA_C 1533 # VA_C 1533 # VA_C 1533 # VA_C 1533 # VA_C 1534 # VA_C 1535 # VA_C 1536 # VA_C 1536 # VA_C 1537 # VA_C 1537 # VA_C 1538 # VA_C 1539 # VA_C 1530 # VA_C 1531 # VA_C 1531 # VA_C 1533 # VA_C 1533 # VA_C 1534 # VA_C 1535 # VA_C 1536 #		
VA_LA-K[] 1517 # VA_LA-K[]-1 1518 # VA_LA-K[]-1 1519 # VA_LA-AND.LC 1520 # VA_LA.AND.NOT.K[] 1521 # VA_LB+D.OXT 1522 # VA_Q 1524 # VA_Q 1524 # VA_Q 1524 # VA_Q+D 1525 # VA_Q+LB 1527 # VA_Q+LB 1527 # VA_Q+LB 1528 # VA_Q+LB 1529 # VA_Q+LB 1530 # VA_Q+C 1530 # VA_Q-C 1531 # VA_Q-C 1531 # VA_Q-C 1531 # VA_Q-C 1531 # VA_Q-C 1533 # VA_Q-C 153		
VA_LA-KI]-1	• • • • • • • • • • • • • • • • • • • •	
VA_LA-Q		
VA_LA.ANDNOT.KEJ 1520 # VA_LA.ANDNOT.KEJ 1521 # VA_LB+D.OXT 1522 # VA_C 1523 # VA_Q 1524 # VA_Q+D 1525 # VA_Q+BD 1525 # VA_Q+KEJ 1526 # VA_Q+LB 1527 # VA_Q+LB 1529 # VA_Q+LC 1529 # VA_Q+C 1530 # VA_Q+C 1531 # VA_Q-KEJ 1531 # VA_Q-KEJ 1533 # VA_RCEJ 1533 # VA_RCEJ 1535 # VA_VA+4 1536 # WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809		
VA_LA.ANDNOT.KC] 1521		
VA_LB+D.OXT		
VA_C VA_Q VA_Q VA_Q+C V		
VA_Q		
VA_Q+D VA_Q+KCJ VA_Q+KCJ VA_Q+LB VA_Q+LB VA_Q+LB,FC VA_Q+LC VA_Q+CC VA_Q+CC VA_Q+CCJ VA_Q+CCJ VA_Q+CCJ VA_Q+CD VA_Q+CD VA_Q+CD VA_Q+CD VA_Q+CD VA_Q+CD VA_Q+CD VA_Q+CD VA_CCD VA_		
VA_Q+K[] 1526 # VA_Q+LB 1527 # VA_Q+LB.FC 1528 # VA_Q+LC 1529 # VA_Q+C 1530 # VA_Q-K[] 1531 # VA_Q-K[] 1531 # VA_Q-K[] 1532 # VA_Q-ANDNOT.KE] 1533 # VA_RC[] 1534 # VA_RC[] 1535 # VA_RC[] 1536 # WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809		
VA_Q+LB 1527		
VA_G+LR.FC		
VA_Q+LC		
VA_Q+FC 1530 # VA_Q-KCJ 1531 # VA_Q-KCJ 1531 # VA_Q-ANDNOT.KCJ 1533 # VA_RCCJ 1534 # VA_RCJ 1535 # VA_RCJ 1536 # WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809		
VA_Q-K[] 1531 # VA_Q-LB 1532 # VA_Q.ANDNOT.K[] 1533 # VA_RCI 1534 # VA_RCI 1535 # VA_RCI 1536 # WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809		
VA_Q-LB		
VA_RCE] 1534 # VA_RCE] 1535 # VA_VA+4 1536 # WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809		1532 #
VA_REIJ 1535 # VA_VA+4 1536 # WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809	VA_Q.ANDNOT.KEJ	1533 #
VAVA+4 1536 # WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809	VA_RCE3	1534 #
WORD 1630 # WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809	VA_RED	1535 #
WRITE.DEST 1631 # WRITE.G.DEST 1632 # Z? 1724 # 1809	VAVA+4	1536 #
WRITE.G.DEST 1632 # 27 1724 # 1809	WORD	1630 #
2? 1724 # 1809	WRITE.DEST	1631 #
	WRITE.G.DEST	1632 #
ZONED? 1725 #	Z?	
	ZONED?	1725 #

NEWSAM.MCR

MICRO2 1L(02) 18-JAN-82 16:15:06. Cross Reference Listins - Expression Names

; NEWSAM.MCR			ICRO2 1L ocation /			32 16:1 dex	5:06		fase	63
£Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F		
U 0000 - 1BFF U 1000 U 1008	Unused 1784= 1771	1790= 1777	1812= 1822=	1816= 1827=	1764 1809	1767 1838	1796=	1799= 1833=		

NEWSAM.MCR MICRO2 1L(02) 18-JAN-82 16:15:06 Page 64
U-code Microword Summary

BSERCH Words not 1000-1FFF in bounds OBSERCH 15 0
Used 15 0
Remainins 1009

Total microwords used in memory U: 15 Total microwords remaining in memory U: 1009 Highest address used in memory U: 1COF (hex)
 NEWSAM.MCR
 MICRO2 1L(02)
 18-JAN-82 16:15:06
 Page 65

 Formula of the property of t

Pass 1 warnings: 0 Pass 2 warnings: 0 Pass 1 errors: 0 Pass 2 errors: 0

B.3 THE OBJECT FILE (.ULD)

```
FRTOL
 RADIX 16
E1C04]=0000003C19C0FA1014047C05
E1C05]=0800003C0180FA0000001C08
 E1C083=0001003C0180FB0800001C09
E1C09J=005C171401C0F80040001C00
E1C00J=00192E2400C0F90802001C06
E10013=001800381980F80440500062
E10063=001000200180421000101000
E10073=0000003C0180F800000004F8
E1C0CJ=001101000180F88800101C02
 E1C023=00001B3C0180F80000001C0A
 E1C03J=00001R3C0580F80014047C0A
 E1C0AJ=0019201411C0FA8800001C0D
E1C0B]=0019200011C0FA9000001C09
E1C0F]=C001203C0180FA8C40500062
E1C0D]=004D371401C0F80040001C00
 FIELD ACF=<71:70>
  CONTROL=3
  NOP≔0
  SYNC=1
TRAP=2
 FIELD ACM=<57:55>
  ABORT=1
  POLY.DONE=6
  PWR.UP=0
 FIELD ADS=<47:47>
  IBA=1
  VA≔0
 FIELD ALU=<69:66>
  A≕OF
  A+B=5
  A+B+1=4
  A+B+PSL+C=OB
  A+B.RLOG=6
  A-B=0
  A-8-1=2
  A~B.RLOG=1
  AND=OD
  ANDNOT=9
  B≔0E
  INST.DEP=3
  NOTA= 0A
  OR≔OC
  ORNOT≈7
  XOR≔8
 FIELD AMX=<81:80>
  LA=0
  RAMX≡1
  RAMX.OXT≈3
RAMX.SXT=2
*FIELD BEN=<76:72>
ACCEL=6
  ALU=1B
  ALU1-0=15
```

```
C31=3
 D.BYTES=18
 D3-0=19
DATA.TYPE=8
DECIMAL=0F
 EALU=12
 END.DF1=8
 IB.O=5
IB.TEST=OB
 INTERRUFT=0E
 IR2-1=9
IRC.ROM=4
 LAST.REF=11
MUL=0C
 NOP=0
 PC.MODES=9
 PSL.CC=1A
PSL.MODE=1C
 RET≔0A
 ROR*2
 SC=14
SIGNS=OD
SRC.PC=OA
STATE3-0=17
STATE7-4=16
 TB.TEST=1D
 2=1
FIELD BMX=<84:82>
 кмх≖б
 LB = 3
 LC=4
MASK=0
PACKED.FL=2
PC=5
 PC.OR.LB=1
 RBMX#7
FIELD CCK=<22:20>
 C_AMXO=6
INST.DEP=7
 LOAD.UBCC=1
 NOP=0
NZ_ALU.VC_0=5
NZ_ALU.VC_VC=6
 N_AMX.Z.TST.UC_UC=3
 ROR≈4
 SET.V=2
FIELD CID=<45:42>
 ACK#5
 CONT=7
 NOF=1
READ.KMX=0B
READ.SC=9
 WRITE.KMX=OF
WRITE.SC=OD
FIELD DK=<91:88>
ACCEL=0A
 BYTE.SWAP=OB
 CLR=OF
 DAL.SC=OD
DAL.SV=OE
 DIV=4
 LEFT=5
LEFT2=1
```

```
NOP=0
 Q=OC
 RIGHT=6
RIGHT2=2
SHF=8
SHF=8
SHF.FL=9
FIELD DT=<79:78>
BYTE=2
INST.DEP=3
  LONG=0
WORD=1
FIELD EALU=<15:13>
 A=0
  A+1=6
  A+B=4
  A-B=5
  ANDNOT=2
 B=3
 NABS.A-B=7
OR=1
FIELD EBMX=<19:18>
AMX.EXP=2
  FE=0
  KMX=1
  SHF.VAL=3
FIELD FEK=<24:24>
LOAD=1
 NOP=0
FIELD FS=<42:42>
 CID=1
 MCT=0
FIELD IBC=<95:92>
BDEST=7
  CLR.O=OC
 CLR.0-3=0E
CLR.0-1=4
CLR.1=0D
CLR.1-5.COND=0F
CLR.2-3=5
  FLUSH=2
NOP=0
START=3
STOP=1
FIELD ID.ADDR=<63:58>
 ACC.0=14
ACC.1=15
ACC.2=16
ACC.CS=17
  CES#OC
  CLK.CS=OA
  COMP=1C
  D.SV=2E
  DAY.TIME=1
 ESP=29
FAULT=1B
  FPDA=2D
  IBUF =0
  INTERVAL=OB
 ISF=2C
KSF=28
MAINT=1D
 NXT.PER=9
POBR=24
```

```
POLR=3C
 FIBR=25
FILR=3D
FARITY=1E
FCBB=3A
FSL=0F
  Q.SV=2F
  RXCS≈4
  RXDB≔5
  SBI.ERR=19
 SBR=26
SCBB=3B
  STL0=18
  STREOF
  SLR=3E
  SSP#2A
  SYS.ID=3
  TO=30
  T1=31
  T2=32
T3=33
T4=34
T5=35
T6=36
  T7=37
  T8=38
T9=39
  TBERO=12
  TBER1=13
TBUF=10
  TIME.ADDR=1A
 TXCS=6
FXDR=7
URREAK=21
USF=2B
USTACK=20
  VECTOR≔OD
WCS.ADDR=22
WCS.DATA=23
FIELD IEK=<31:30>
EACK=3
IACK=2
  ISTR=1
 NOP #0
ADDRESS J=<12:0>
  INT.B=4F8
  IRD=62
  SRCH=1CO4
 SRCH=1C04

SRCH.1=1C09

SRCH.2=1C00

SRCH.3=1C06

SRCH.4=1C0C

SRCH.5=1C0A

SRCH.6=1C0D
FIELD KMX=<63:58>
 .1=1
.10=19
  .14≔8
  .18=1F
  .19≕2E
  .1A≔39
  .1B=3B
.1E=14
```

```
.1F=23
 .1F00=24
.2=2
.20=1D
.24=3A
.28=0B
  .3=3
  .30=1E
  .3030=32
  .34=0A
 .3F=15
.3FF=20
 .4=4
.40=0C
  .4000=2C
  .50=0D
  .6≖35
  .60=29
  .7=17
 .7C=27
.7E=3E
.7F=16
.7FF0=0E
.8=0
  .80=10
  .8000=11
  .88=31
  .9=36
  .A≔30
  .60≈9
  .BO=25
  .C=21
.C0=34
  .D=22
  .DFCF=2B
  .E003=26
  .EF≡0F
  •F=18
  .FO=33
 .FF=12
.FF00=13
.FFE0=28
.FFE8=1A
  .FFFO=1B
  .FFF1=2D
 .FFF5=38
 .FFF6=37
  .FFF8=1C
  .FFF9≔2F
 .FFFC=3C
 SC=7
SC=7
SP1.CON=5
SP2.CON=6
ZERO=6
FIELD MCT=<47:42>
ALLOW.IB.READ=3E
EXTWRITE.P=28
 INVALIDATE=24
 LOCKREAD.P=3A
 LOCKREAD.V.NOCHK=1A
 LOCKREAD.V.WCHK=1C
LOCKWRITE.F=2E
```

```
LOCKWRITE.V.XCHK=OE
 MEM.NOP=2
 READ.INT.SUM=36
 READ.INT.SUM=36
READ.P=32
READ.V.IBCHK=16
READ.V.NEWPC=18
READ.V.NOCHK=12
READ.V.RCHK=10
READ.V.WCHK=14
SBI.HOLD=20
SBI.HOLD+UNJAM=22
TEST.RCHK=0
TEST.WCHK=4
VALIDATE=26
 VALIDATE=26
WRITE.P=2A
 WRITE.V.NOCHK=OA
WRITE.V.WCHK=OC
FIELD MSC=<29:26>
CHK.CHM=1
 CHK.FLT.OFR=2
 CHK.ODD.ADDR=3
 CLR.FFD=8
CLR.NEST.ERR=0A
INH.CM.ADDR=0F
  IRD=4
 LOAD.ACC.CC=6
 LOAD.STATE=5
 NOP=0
 READ.RLOG=7
 RETRY.NO.TRAP=OD
 RETRY.TRAP=OE
 SECOND.REF=OC
SET.FPD=9
SET.NEST.ERR=0B
FIELD PCK=<34:32>
NOF=0
 PC+1=4
 PC+2=5
 PC-14=6
 FC+N=7
FC_IBA=2
FC_VA=1
 VA+4=3
FIELD QK=<54:51>
 ACCEL=0₽
 CLR=OF
 D≈oc
 DEC.CON=OA
  t D=OE
 LEFT=5
LEFT2=1
 NOP=0
 RIGHT=6
 RIGHT2=2
 SHF#8
 SHF.FL=9
F16LD RAMX=<77:77>
 \mathbf{D} = \mathbf{0}
FIELD RBMX=<77:77>
 p = 1
 Q==O
FIELD SCK=<23:23>
```

```
LOAD=1
NOP=0
FIELD SGN=<50:48>
 ADD.SUB=6
 CLR.SD+SS=7
 LOAD.SS=1
 N0F=0
NOF=0
NOT.SD=3
SD.FROM.SS=4
SS.FROM.SD=2
SS.XOR.ALU=5
FIELD SHF=<87:85>
 AL.U≔0
 ALU.DT=3
 LEFT=1
 LEFT3=5
 RIGHT=2
 RIGHT2=4
FIELD SI=<57:55>
 ASHL=2
 ASHR=1
 DIV=5
 0.140 \pm 0
 MUL+≕6
 MUL.--=7
ZERO=3
FIELD SMX=<17:16>
ALU=2
 ALU.EXP=3
 EALU=0
 FE=1
FIELD SP0=<41:35>
 LOAD.LC.SC=6
NOP=0
WRITE.RC.SC=7
FIELD SPO.AC=<41:38>
LOAD.LA=2
 LOAD.LAB=1
 WRITE.RAB=3
FIELD SPO.ACN=<37:35>
 PRN#3
 PRN+1=4
SC=5

SF1+1=6

SF1.SF1=0

SF2.SF1=2

SP2.SF2=1

FIELD SF0.ACN11=<37:35>
 DST.DST=1
 DST.SRC=2
 SC=5
 SRC.OR.1=4
 SRC.SRC=0
FIELD SP0.R=<41:39>
 LOAD.LAB=4
 LOAD.LAB1.WRITE.RC=6
 LOAD.LC=2
LOAD.LC.WRITE.RAB1=7
WRITE.RAB=5
 WRITE . RC=3
FIELD SPO.RAB=<38:35>
 AF=OC
FF=OD
```

```
RO=0
R1=1
R15=0F
R2=2
R3=3
R4=4
R5=5
R6=6
R7=7
SP=0E
FIELD SPD.RC=<38:35>
LC.SV=8
MBIT.VA=0F
PC.SV=0C
PTE.MASK=0F
PTE.PA=0B
PTE.PA=0B
PTE.VA=0A
SC.SV=0D
TO=0
T1=1
T2=2
T3=3
T4=4
T5=5
T6=6
T7=7
VA.REF=0E
VA.SV=9
FIELD SUB=<65:64>
CALL=1
NOP=0
RET=2
SPEC=3
FIELD VAK=<25:25>
LOAD=1
NOP=0
END
```

APPENDIX C

SAMPLE MICROPROGRAM FOR SYSTEM REVISION < 7

This appendix contains a sample VAX 11/780 microprogram, which performs an unsigned binary search on a vector of longwords in main memory. The parameters of the routine, the value to be searched for and the beginning and end of the vector, are passed in registers.

A command file that assembles, loads, and executes this sample microprogram is provided in the VAX 11/780 WCS kit. To invoke this file in the VMS environment, type:

@[SYSEXE]WCSTOLTST

This command file assembles the input listing (Section C.1) and produces the listing file (Section C.2) and the object file (Section C.3) which are written to [VAXWCSTOL]SAMPLE.MCR and [VAXWCSTOL]SAMPLE.ULD. It then loads the object file into the extended WCS and runs the test program BSTEST (Appendix D). BSTEST executes an XFC instruction, which causes the sample microprogram loaded in the WCS to be executed. If the microprogram executes properly, BSTEST prints the following message on the terminal:

[&]quot;Successful Test Completion"

C.1 THE INPUT FILE (.MIC)

.TOC 'Binary search routine'

.BOUNDS/BSERCH:1400,17FF

.REGION /1400,17FF

```
for the U-code microword summary page
                                                          ; and names the report boundary BSERCH.
Sample microcode to perform an unsided binary search through
f a vector of alianed longwords in main memory.
          RO - Search comparand. Routine succeeds by finding a
                      memors cell containing same data as RO.
          R1 - Lower address bound. Aligned longword address of
 lowest address of vector to be searched.

R2 - Upper address bound. Alianed longword address of highest address of vector to be searched.

It is implied that R1 lssu R2, and that the memory between the
  addresses in R1 and R2 contains a sorted vector, in ascending
; unsigned order.
  Outputs if search finds a match.
          ts it search finds a match.

CC<Z> - Clear

RO - Search comparand.

R1 - Match address. Address of longword containing same data as RO.

R2 - Used by search for temporary address values.
  Outputs if search does not find a match.
          CC<Z> - Set
RO - Search comparand.

    Used by search for temporary address values.
    Used by search for temporary address values.

          R1
          R2
```

fUser wcs space.

This defines the report boundries

SRCH:;-----; Q_RCR2J;;GET UPPER BOUND ADDR TO Q STATE_KCZEROJ;INITIALIZE STATE REGISTER

D_RCROJ;GET COMPARAND TO HOLD IN RC

ALU_D, PREPARE TO WRITE COMPARAND TO RC LAB_R1&RC[T1]_ALU; WRITE COMPARAND, GET LOWER BOUND

SRCH.1:;----;
Q_(LA+Q).RIGHT,;COMPUTE MIDPOINT ADDRESS
INTRPT.STROBE,;TEST FOR INTERRUPT REQUESTS
STATEO?;IS IT TIME TO STOP?

=0
SRCH.2:;0-----;STATEO=0. KEEP LOOKING FOR MATCH.
Q_Q.ANDNOT.KC.33,;FORCE LONGWORD ALIGNMENT
VA_ALU,;GET READY TO READY MIDPOINT OF VECTOR
LC_RCCT13,;LATCH COMPARAND INTO LC
INT?,J/SRCH.3;IS THERE AN INTERRUPT REQUEST?

;1------;STATEO=1. SEARCH FAILED. NO MATCH.
ALU_KIZEROJ;;
CCK/NZ_ALU.VC_0;LONG;;RETURN Z=1 TO FLAG FAILURE.
CLR.IB.OPC;PC_PC+1;#MOVE ON TO THE NEXT INSTRUCTION
J/IRD;

=110
SRCH.3:;1110------;NO INTERRUPT REQUESTS
DLLONGJ_CACHE,;READY MIDPOINT ENTRY OF VECTOR
ALU_RER2J.XOR.Q,;COMPARE MIDPOINT EQL UPPER BOUND
CLK.UBCC,J/SRCH.4;

;111-----;INTERRUFT REQUEST IS UP J/INT.B;TAKE IT. RESUME FROM REG'S AS IS.

```
# WE HAVE ALSO SET THE MICROBRACH Z BIT ACCORDING TO A COMPARE OF THE MEMORY ADDRESS WITH THE CURRENT UPPER BOUND. IF THEY ARE # EQUAL, THIS IS THE LAST POSSIBLE COMPARISON. A MATCH FAILURE # HERE IMPLIES THAT THERE IS NO MATCH TO BE FOUND.
ALU_D-LC, COMPARE MEMORY TO COMPARAND
LONG,CLK,UBCC, FRECORD COMPARE RESULT
LA_RACR1], FLATCH LOWER BOUND INTO LA (LB HAS ???
Z?FIS MIDPOINT EQL UPPER BOUND?
=0;0------;ALU Z=0. NOT END OF SEARCH ALU?,J/SRCH.5;TEST RESULT OF COMPARE
#1----- END OF SEARCH
STATE_KE.13,; SET STATEO TO MARK END OF SEARCH.
ALU? CHECK FOR LAST CHANCE MATCH
SRCH.5: #1010------#ALU Z=0, C=1. RO GTRU MEM
Q_Q+K[.4], #LOWER LIMIT MUST BE GREATER THAN THIS RCR1]_ALU, FREMEMBER IN R1.
J/SRCH.6#
$1011------ RO LSSU MEM
Q_Q-KE.4], UPPER LIMIT MUST BE LESS THAN THIS
RER23_ALU, FREMEMBER IN R2
J/SRCH.11GO TRY AGAIN
=1111;1111------;ALU Z=1, C=1. RO EQL MEM
RERIJ_Q, FOUND IT!
CCK/NZ_ALU,VC_O,LONG,#SET Z=O TO INDICATE MATCH
CLR.IB.OPC,PC_PC+1,#GO TO NEXT INSTRUCTION
J/IRD
Q_(Q+LB).RIGHT, COMPUTE NEW MIDPOINT, LOOP
INTRPT.STROBE,;
STATEO?, J/SRCH.2; CHECK FOR END, LOOP
DEFINE LABLES TO INTERFACE WITH PCS
0062:IRD:
04F8:INT.B:
```

C.2 THE LISTING FILE (.MCR)

ş	; OLDSAM.MCR ;		MICRO2 1L(02) 18-JAN-82 16:19:40 Table of Contents				Pas	
	2	Machine definition	:	Cont	rol word	chart		
-	56	Machine definition				ALU, AMX		
	97	Machine definition		BEN,				
-	150	Machine definition			CID, DK,	nT		
-	205	Machine definition				EK, FS, IEK,	TRC	
	255	Machine definition			DDR, J	L	120	
-	330	Machine definition		KMX				
	405	Machine definition		MCT,	MSC			
	452	Machine definition			QK, RAMX	. PRMY		
	487	Machine definition				SI, SMX		
-	529	Machine definition					.ACN11, SPO.R	
	568	Machine definition				RC, SUB, VAK	·WCHII, SLO·K	
			-					
	617	Machine definition	-		dity chec			
ŷ	624	Macro definition	:	Resi	ster trar	isfer macros		
ŷ	1538	Macro definition	:	Non-	transfer	macros		
ŷ	1634	Macro definition	:	Bran	ch enable	macros		
8	1729	Binary search routine	2					

OLDSAM.MCR # VAXDEF.MIC MICRO2 1L(02)

18-JAN-82 16:19:40

Fase

#1 .NOLIST #1728 .L.IST #Inhibit listing for VAXDEF.MIC

```
# OLDSAM.MCR
# BSERCH.MIC
                                                                                                                                                            MICRO2 1L(02) 18-
Binary search routine
                                                                                                                                                                                                                                                            18-JAN-82 16:19:40
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        Page
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         39
                                                                                                                                                                                                                                                                                                           .TOC "Binary search routine"
.REGION /1400,17FF
.BOUNDS/BSERCH:1400,17FF
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          iUser was space.
IThis defines the report boundries
Ifor the U-code microword summary page
I and names the report boundary BSERCH.
                                                                                                                                                                                                                       #1730
#1731
                                                                                                                                                                                                                        $1732
$1733
$1734
                                                                                                                                                                                                                       #1735
#1736

† Sample microcode to perform an unsided binary search through
† a vector of aligned longwords in main memory.

                                                                                                                                                                                                                       #1737
#1738
                                                                                                                                                                                                                                                                  # INPUTS
                                                                                                                                                                                                                                                                # INPUTS
# RO - Search comparand. Routine succeeds by finding a
# memory cell containing same data as RO.
# R1 - Lower address bound. Alianed longword address of
# lowest address of vector to be searched.
# R2 - Upper address bound. Alianed longword address of
# highest address of vector to be searched.
# It is implied that R1 Issu R2, and that the memory between the
                                                                                                                                                                                                                       #1738
#1739
#1740
#1741
#1742
#1743
#1744
                                                                                                                                                                                                                       $1746
$1747
$1748
                                                                                                                                                                                                                                                                            addresses in R1 and R2 contains a sorted vector, in ascending
                                                                                                                                                                                                                                                                  ; unsidned order.

// Outputs if search finds a match.
// CC<Z> - Clear
// RO - Search comparand.
// R1 - Match address. Address of longword containing same data as RO.
// R2 - Used by search for temporary address values.
// R2 - Used by search for temporary address values.
// R3 - Used by search for temporary address values.
// R3 - Used by search for temporary address values.
// R3 - Used by search for temporary address values.
// R4 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by search for temporary address values.
// R5 - Used by S6 - Use
                                                                                                                                                                                                                       $1749
$1750
$1751
                                                                                                                                                                                                                       #1752
#1753
                                                                                                                                                                                                                      $1754
$1755
$1756
$1756
$1757
                                                                                                                                                                                                                                                                 # Outputs if search does not find a match.
                                                                                                                                                                                                                                                                                                            CC<Z> - Set
RO - Search comparand.
                                                                                                                                                                                                                       $1758
$1759
$1760

    Used by search for temporary address values.
    Used by search for temporary address values.

                                                                                                                                                                                                                                                                                                            R2
```

	BAM.MCR RCH.MIC	MICRO2 1L Binary sea			2 16:19:40		Page	40
			#1761 #1762	SRCH:	1	-4		
			\$1763		Q_R(R23)	JGET UPPER BOUND ADDR TO Q		
11 140	4, 0000,003C,19C0,FA10,	1404.7405	11764		STATE_KCZERO)	JINITIALIZE STATE REGISTER		
U 140	47 00007000C717C071 H107	1170717705	\$1765		OTH TELECTOR	TANTIALIZE STATE RESISTER		
			1766		!	1		
U 140	5, 0800,003C,0180,FA00,	0000,1408	#1767		D_RCROJ	JGET COMPARAND TO HOLD IN RC		
			11768					
			\$1769		;	-)		
			#1770		ALU_D,	*PREPARE TO WRITE COMPARAND TO RO	;	
U 1408	3, 0001,003C,0180,FB08,	0000,1409	#1771		LAB_R1&RC[T1]_ALU	#WRITE COMPARAND, GET LOWER BOUND)	
			#1772					
			11773					
					;	-1		
			11775		Q_(LA+Q).RIGHT,	COMPUTE MIDPOINT ADDRESS		
			#1776		INTRFT.STROBE,	ITEST FOR INTERRUPT REQUESTS		
IJ 1409	P, 005C,1714,01CO,F800,	4000,1400	\$1777		STATEO?	IS IT TIME TO STOP?		
			#1778	_				
				=0				
						-#STATEO=0. KEEP LOOKING FOR MATC	:н.	
			1781		Q_Q.ANDNOT.KC.33,	FORCE LONGWORD ALIGNMENT		
			#1782 #1783		VA_ALU, LC_RCCT13,	#GET READY TO READY MIDPOINT OF V	ECTOR	
	, 0019,2E24,0DC0,F908,	0000-1407	11783 11784		INT?,J/SRCH.3			
U 1400), UU19,2E24,UDCU,F9U8,	020011408	\$1785		INI FFJ/ SKUN-3	115 THERE AN INTERRUPT REQUEST!		
			#1786		! 1	-#STATEO=1. SEARCH FAILED. NO MAI	reu.	
			#1787		ALU_K[ZERO],	*	Cn.	
			\$1788		CCK/NZ_ALU.VC_O,LONG,	FRETURN Z=1 TO FLAG FAILURE.		
			11789		CLR.IB.OPC.PC_PC+1,	*MOVE ON TO THE NEXT INSTRUCTION		
11 1401	l, C018,0038,1980,F804,	4050.0042	11790		J/IRD	1		
., 140.	., .010,0000,1,00,,00,,	103070002	11791		J/ IKD	•		
				=110				
					#110	-ING INTERRUPT REQUESTS		
			\$1794		DELONG3_CACHE,	#READY MIDFOINT ENTRY OF VECTOR		
			1795		ALU_RER21.XOR.Q,	#COMPARE MIDPOINT EQL UPPER BOUNT)	
U 140	6, 001C,0020,0180,4210,	0010,140C	¥1796		CLK.UBCC, J/SRCH.4	,		
			11797					
			# 1798			- INTERRUPT REQUEST IS UP		
U 1407	, 0000,003C,0180,F800,	0000,04FB	£1799		J/INT.B	JTAKE IT. RESUME FROM REG'S AS I	S.	

41

```
# OLDSAM.MCR
# BSERCH.MIC
                                      MICRO2 1L(02) 18-.
Binary search routine
                                                              18-JAN-82 16:19:40
                                                                                                                                                      Pase
                                                               # WE HAVE ALSO SET THE MICROBRACH Z BIT ACCORDING TO A COMPARE OF THE MEMORY ADDRESS WITH THE CURRENT UPPER BOUND. IF THEY ARE EQUAL, THIS IS THE LAST POSSIBLE COMPARISON. A MATCH FAILURE HERE IMPLIES THAT THERE IS NO MATCH TO BE FOUND.
                                                     #1800
                                                     11801
                                                     #1802
                                                     £1803
                                                     #1804
                                                     11805
                                                     $1806
$1807
                                                                          ALU_D-LC,
LONG,CLK.UBCC,
                                                                                                          #COMPARE MEMORY TO COMPARAND #RECORD COMPARE RESULT
                                                     1808
                                                                          LA_RACR13,
                                                                                                          $LATCH LOWER BOUND INTO LA (LB HAS ??? $1S MIDPOINT EQL UPPER BOUND?
U 140C, 0011,0100,0180,F888,0010,1402
                                                     $1809
$1810
                                                      1811
                                                                                                         - JALU Z=O. NOT END OF SEARCH
JTEST RESULT OF COMPARE
U 1402, 0000,1B3C,0180,F800,0000,140A
                                                                          ALU?,J/SRCH.5
                                                     #1812
                                                                                        $1814
$1815
                                                                           STATE_KC.13,
U 1403, 0000,1B3C,0580,F800,1404,740A
                                                     #1816
                                                     $1817
                                                                =1010
                                                                SRCH.5: #1010-----
                                                                                                         - JALU Z=0, C=1. RO GTRU MEM JLOWER LIMIT MUST BE GREATER THAN THIS
                                                     #1819
                                                                          Q_Q+KE.41,
                                                     11820
                                                                          RERIJ_ALU,
J/SRCH.6
                                                     1821
                                                                                                           FREMEMBER IN R1.
U 140A, 0019,2014,11C0,FA88,0000,140D
                                                     #1822
                                                     11823
                                                     1824
                                                                                                ------;ALU Z=0, C=0. RO LSSU MEM
;UPPER LIMIT MUST BE LESS THAN THIS
;REMEMBER IN R2
                                                                          Q.Q-KC.43,
                                                     #1825
                                                     11826
                                                                          RER23_ALU,
U 140B, 0019,2000,11C0,FA90,0000,1409
                                                     #1827
                                                                           J/SRCH.1
                                                                                                           ∮GO TRY AGAIN
                                                     11828
                                                                          #1829
#1830
                                                                =1111
                                                                          RER13_Q,
CCK/NZ_ALU.VC_O,LONG,
CLR.IB.OPC,PC_PC+1,
                                                     1831
                                                     £1832
U 140F, C001,203C,0180,FA8C,4050,0062
                                                     1833
                                                     11834
                                                     11835
                                                                SRCH.6: #----
                                                     #1836
#1837
                                                                          Q_(Q+LB).RIGHT,
INTRPT.STROBE,
                                                                                                           COMPUTE NEW MIDPOINT, LOOP
U 140D, 004D, 3714, 01CO, F800, 4000, 1400
                                                                          STATEO?, J/SRCH.2
                                                                                                          JCHECK FOR END, LOOP
                                                     ;1838
                                                     #1839
#1840
                                                                # DEFINE LABLES TO INTERFACE WITH PCS
                                                                          IRD:
INT.B:
                                                     £1842
                                                                04F8:
```

OLDSAM.MCR MICRO2 1L(02) 18-JAN-82 16:19:40 and Defined Values

J 326 *
INT.B 1799 1842 *
IRD 1790 1833 1841 *
SRCH 1762 *
SRCH.1 1774 * 1827 SRCH.2 1780 * 1838 SRCH.2 1780 * 1838 SRCH.3 SRCH.4 1794 1793 *
SRCH.4 1794 1793 *
SRCH.5 SRCH.5 1812 1819 *
SRCH.6 1822 1835 *

; OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40
¢ GEBSHITTIGK	Cross Reference Listins - Macro Names
AC.LOW? .	1636 #
ACC.SYNC?	1637 #
ACCEL?	1638 #
ALIGNED?	1639 #
ALU.N?	1640 #
	1641 #
ALU1-0?	1642 # 1812 1816
ALU? ALU1	626 #
	627 #
ALU_O(A)	628 #
ALU_O+D	629 #
ALU_0+D+1	630 #
ALU_O+KEJ	631 #
ALU_0+KC3+1	
ALU_0+LB+1	632 #
ALU_O+LC	633 #
ALU_0+LC+1	634 #
ALU_O+MASK+1	635 #
ALU_0+Q	636 #
ALU_0+Q+1	637 #
ALU_O-D	638 #
ALU_0-D-1	639 #
ALU_O-KEJ	640 #
ALU_0-K[]-1	641 #
ALU_O-LB	642 #
ALU_O-LC	643 #
ALU_0-LC-1	644 #
ALU_O-Q	645 #
ALU_0-Q-1	646 #
ALU_OCID_	647 #
ALU_0E3LC	648 #
ALULD	649 # 1770
ALU_D(B)	650 + 651 +
ALU_D+KEJ	652 *
ALU_D+KEJ+1	653 ‡
ALU_D+K[].RLOG	654 *
ALU_D+LB	455 #
ALU_D+LC ALU_D+LC+1	656 #
	457 #
ALU_D+LC+PSL·C ALU_D+Q	658 #
ALU_D+Q+1	659 #
ALU_D+Q+PSL+C	660 #
ALU_D+RLOG	661 #
ALULD-KCJ	462 #
ALU_D-KCJ-1	663 #
ALU_D-LB	664 #
ALU_D-LB.RLOG	665 #
ALU_D-LC	666 # 1806
ALU_D-LC-1	667 #
ALU_B-Q	668 #
ALU_D-Q-1	669 #
ALU_D.OXTC3	670 #
ALU_D.OXTE3+KE3	671 *
ALU_D.OXTC3+LC	672 #
ALU_D.OXTE3+CC	673 #
a sensor on At T M/S f Se of C M	

OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names	Pase	44
ALU_D.OXTEJ-KEJ	674 🛊		
ALU_D.OXTEJ-Q	675 #		
ALU_D.OXTEJ.AND.KEJ	676 #		
ALU_D.OXT[].ANDNOT.K[]	677 #		
ALU_D.OXTCJ.OR.Q	678 #		
ALU_D.AND.KC]	679 \$		
ALU_D.AND.MASK	680 #		
ALU_D.ANDNOT.KEJ	681 #		
ALU_D.ANDNOT.MASK	682 #		
ALU_D.ANDNOT.Q	683 *		
ALU_D.OR.KEJ	684 #		
ALU_D.OR.LC	685 #		
ALU_D.OR.Q	686 #		
ALU_D.OR.RCEJ	687 ‡		
ALU_D.ORNOT.MASK	688 #		
ALU_D.SXTE3	689 #		
ALU_D.SXTCJ+KCJ	690 #		
ALU_D.SXTEJ+Q	691 🛊		
ALU_D.SXTEJ.AND.KEJ	693 #		
ALU_D.SXT[].ANDNOT.K[]	692 #		
ALU_D.XOR.KE3	694 #		•
ALU_D.XOR.LC	695 * 696 *		
ALU_D.XOR.Q	697 #		
ALU_D.XOR.RC[] ALU_D.XOR.R[]	698 #		
ALU_DCJKCJ	699 #		
ALU_DCJLB	700 #		
ALU_DE JLC	701 #		
ALU_DEJQ	702 #		
ALU_KEJ	703 # 1787	•	
ALU_LA	704 #		
ALU_LA+K[]	705 #		
ALU_LA+K[]+1	706 🛊		
ALU_LA+KEJ.RLOG	707 #		
ALULLATLB	708 #		
ALU_LA+LC	709 #		
ALU_LA+LC+1	710 * 711 *		
ALU_LA+LC+PSL+C	711 * 712 *		
ALU_LA+Q ALU_LA-D	713 #		
ALU_LA-D-1	714 #		
ALU_LA-K[]	715 #		
ALU_LA-KCJ-1	716 #		
ALU_LA-K[].RLOG	717 #		
ALU_LA-LC	718 #		
ALU_LA-Q	719 🛊		
ALU_LA-Q-1	720 #		
ALU_LA.AND.KEJ	721 🛊		
ALU_LA.AND.LC	722 🛊		
ALU_LA.ANDNOT.KEJ	723 #		
ALU_LA.ANDNOT.MASK	724 #		
ALU_LA.OR.K[]	725 ‡		
ALU_LA.XOR.LC	726 #		
ALU_LACID	727 #		
ALU_LACJLB	728 #		

Pase 45

; OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40
ŷ	Cross Reference Listins - Macro Names
ALULLACOQ	729 #
ALU_LB	730 #
ALU_LC	731 *
ALU_NOT.D	732 *
ALU_NOT+KEJ	733 #
ALU_NOT.RCEI	734 #
ALU_PACK.FP	735 #
ALU_PC	736 #
ALU_Q	737 #
ALU_Q(B)	738 #
ALU_Q+KEJ	739 #
ALU_Q+KEJ+1	740 #
ALU_Q+LB	741 #
ALU_Q+LB+1	742 #
ALU_Q+LC	743 #
ALU_Q+LC+1	744 #
ALU_Q+LC+PSL.C	745 #
ALU_Q+MASK	746 *
ALU_Q-D	747 #
ALU_Q-D-1	748 #
ALU_Q-KCJ	749 #
ALU_Q-LB	750 *
ALU_Q-LC	751 *
ALU_Q-MASK-1	752 #
ALU_Q.OXTEJ	753 *
d+EJTXO.9_UJA	754 #
ALU_Q.OXTC3+D+1	755 #
ALU_Q.OXTEJ+KCJ	756 🛊
ALU_Q.OXTC3-D	757 #
ALU_Q.OXTE3-KE3	758 *
ALU_Q.OXTEJ.ANDNOT.KEJ	759 #
ALU_Q.OXTE3.OR.D	761 #
ALU_Q.OXTEJ.OR.KEJ	760 #
ALU_Q.AND.D	762 ¥
ALULQ.AND.KCI	763 #
ALU_Q.ANDNOT.KEJ	764 #
ALU_Q.ANDNOT.MASK	765 #
ALU_Q.ANDNOT.REJ	766 #
ALU_Q.OR.KEJ	767 #
ALU_Q.OR.LC	768 #
ALU_Q.ORNOT.KED	769 #
ALULQ.SXTED	770 #
ALU_Q.SXTED+KED	771 #
ALU_Q.SXTEJ+LB	772 #
ALULQ.SXTED+LB+1	773 #
ALU_Q.SXTCJ+PC	774 #
ALULQ.SXTC3.ANDNOT.KC3	775 #
ALU_Q.XOR.D	776 #
ALU_Q.XOR.KCJ	777 *
ALU_Q.XOR.LC	778 #
ALU_Q.XOR.RCEJ	779 #
ALU_GCID	780 * 701 *
ALULR(DST)	781 #
ALU_R(SC).ANDNOT.KEJ	782 *
ALU_R(SP1)+KEJ.RLOG	783 #

; OLDSAM.MCR	MICRO2 1L(02) Cross Reference	18-JAN-82 16:19:40 Listins - Macro Names	Pa
ALU_RC(SC)	784 #		
ALU_RCE I	785 #		
ALU_RLOG	786 #		
ALU_R[]	787 #		
ALULRES-KES	788 #		
ALU_REJ.AND.KEJ	789 #		
ALU_REJ.AND.LC	790 #		
ALU_REJ.ANDNOT.KEJ	791 #		
ALU_REJ.ANDNOT.MASK	792 # `		
ALU_REJ.OR.KEJ	793 #		
ALU_REJ.ORNOT.KEJ	794 #		
ALU_REJ.XOR.KEJ	795 #		
ALU_RCJ.XOR.Q	796 # 1795		
B.FORK	1540 #		
BCDSGN?	1644 #	1	
BYTE	1541 #		
C.FORK	1543 #		
C31?	1646 #		
CACHE.INVALIDATE	1544 #		
CACHE.P.DE3	798 #		
CACHECILD	799 #		
CACHELD(QUAD)	800 #		
CACHE_D.INST.DEP	801 #		
CACHELDED	802 #		
CACHE_DEJ.LK	803 #		
CACHELDE 1 • NOCHK	804 #		
CALL	1545 #		
CALLEI	1546 #		
CHK.FLT.OPR	1547 #		
CHK.ODD.ADDR	1548 #		
CLK.UBCC	1549 # 1796	1807	
CLR.FFD	1551 #		
CLR.IB.COND	1552 #		
CLR.IB.OPC	1553 # 1789	1832	
CLR.IB.SPEC	1554 #		
CLR.IBO-1	1555 #	,	
CLR.IBO-3	1556 #		
CLR.IB2-3	1557 #		
CLR.IB2-5	1558 #		
CLR.NEST.ERR	1559 #		
CLR.SD&SS	1560 #		
CONSOLE MODE?	1647 #		
D&Q_D+Q	806 # 807 #		
D&RCCI_PC	808 #		
D&VA_ALU D&VA_D+LC	809 #		
D&VA_D+G	810 #		
D&VA_D-KCJ	811 #		
DSVA_LA	812 #		•
D\$VA_LB	813 #		
Dava_Q	814 #		
D&VA_Q+LB.PC	815 #		
D(1)?	1649 #		
D.BO?	1650 #		
D.B1?	1651 #		
## + #P # 1	TOUT #		

Pase 47

DLDSAM.MCR	MICRO2 1L(02) Cross Reference	18-JAN-82 16:19:40 Listins - Macro Names
D.B2?	1652 #	
D.BYTES?	1653 #	
D.NE.0?	1654 #	
DO?	1655 #	
D2-0?	1656 #	
D2?	1657 #	
D3-0?	1658 #	
D31?	1659 #	
1)37	1660 #	
DATA.TYPE? DBL?	1661 # 1662 #	
DCI_CACHE	817 # 1794	1
DEJ_CACHE.IBCHK	818 #	•
DEJ_CACHE.LK	819 #	
DEJ_CACHE · NOCHK	820 #	
DEJ_CACHE • P	821 #	
DED_CACHE.WCHK	822 #	
n_0	824 #	
D_O+KCJ+1	825 #	
D_O+LC+1	826 #	
DO-D	827 #	
D_O-KEJ	828 #	
D_0-Q	829 #	
D_0-Q-1	830 # 831 #	
D_ACCEL*SYNC	832 #	
D_ALU D_ALU(FRAC)	833 #	
D_ALU.LEFT	834 #	
D_ALU.LEFT2	835 #	
D_ALU.LEFT3	836 #	•
D_ALU.RIGHT	837 #	
D_ALU.RIGHT2	838 #	
D_BLANK	839 #	
D_CACHE.INST.DEP	840 #	
D_CACHE.LKCJ	841 #	
D_CACHE.WCHKE3	842 # 843 #	
D_CACHEEJ D_D(FRAC)	844 #	
D_D+KEJ	845 #	
D_D+KEJ+1	846 #	
D_D+LB	847 #	
D_D+LC	848 #	
D_D+LC+PSL+C	849 #	
D_D+Q	850 #	
D_D+Q+1	851 #	
D-D-KEI	852 #	
D_D-LC	853 #	
D_D-Q D_D-Q-1	854 # 855 #	
D_D.OXTC3	856 #	
D_D.OXTE3+KE3	857 #	
D_D.OXTEJ+Q	858 #	
D_D.0XTEJ+0+1	859 #	
D_D.OXTEJ.ANDNOT.KEJ	860 #	
D_D.OXTEJ.OR.Q	861 #	

Pase 48

; OLDSAM.MCR ;	MICRO2 1L(02) Cross Reference	18-JAN-82 16:19:40 Listins - Macro Names
D_D.OXTE3.XOR.Q	862 #	
D_D.OXTEJ.XOR.RCEJ	863 #	
D_D.AND.KE3	864 #	
D_D.AND.KEJ.LEFT2	865 #	
D_D.AND.KCJ.RIGHT	866 🛊	
D_D.AND.LC	867 #	
D_D.AND.MASK	868 #	
D_D.AND.Q	869 #	
D_D.AND.RCCJ	870 #	
D_D.ANDNOT.KCJ	871 #	
D_D.ANDNOT.LC	872 #	
D_D.ANDNOT.FSWZ	873 #	
D_D.ANDNOT.Q	874 #	
D_D.ANDNOT.RCCJ	875 #	
D_D.LEFT	876 #	
D_D.LEFT2	877 #	
D_D.OR.ASCII	878 #	
D_D.OR.KE3	879 #	
D_D.OR.PSWC	880 #	
D_D.OR.PSWV	881 #	
D_D.OR.Q	882 #	
D_D.OR.RCEJ D_D.OR.REJ	883 # 884 #	
D_D.ORNOT.MASK	885 #	
D_D.RIGHT	886 #	
D_D.RIGHT(B)	887 *	
D_D.RIGHT2	888 #	
D_D.SWAF	889 # 890 #	
D_D.SXTE3 D_D.SXTE3.RIGHT	891 #	
D_D.XOR.KEJ	892 #	
D_D.XOR.LC	893 #	
D_D.XOR.Q	894 #	
D_DAL.NORM	895 #	
D_DAL . SC	896 #	
D_DCJKCJ	897 #	
D_DEJMASK	898 #	
D_DC 3Q	899 #	
D_INT.SUM	900 #	
D_KE3	901 #	
D_KCJ.RIGHT	902 #	
D_KCJ.RIGHT2	903 #	
D_LA	904 #	
D_LA(FRAC)	905 #	
D_LA+D+PSL .C	906 #	
D_LA-D	907 #	
D_LA-KEJ	908 #	
D_LA.AND.KEJ	909 #	
D_LA.RIGHT	910 #	
D_LB	911 #	
D_LB.PC	912 #	
D_LC	913 #	
D_LC(FRAC)	914 #	
D_NOT.D	915 #	
D_NOT+KE3	916 #	

; OLDSAM.MCR ;	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names	Page	49
DNOT.MASK	917 #		
DNOT.Q	918 #		
D_NOT.RC3	919 #		
D_PACK.FP	920 *		
D_PACK.FP.LEFT	921 #		
	922 #		
D_PC	923 #		
D_PC.LEFT	924 *		
D_Q	925 #		
D_Q(FRAC)	926 *		
D_Q+D	927 *		
D_Q+KE3	928 *		
D_Q+LB	929 #		
D_Q+FC	930 #		
D_0-D			
n_q-n-1	931 ‡ 932 ‡		
D_Q-KC]	933 *		
D_Q-K[]-1	934 #		
D_Q-PCSV	935 #		
D_Q.OXTEJ	936 #		
D_Q.AND.KCJ D_Q.AND.LC	937 #		
D_Q.AND.MASK	938 #		
	939 #		
D_Q.AND.RCEJ D_Q.ANDNOT.D	940 #		
D_Q.ANDNOT.KCJ	941 #		
D_Q.ANDNOT.MASK	942 *		
D_Q.ANDNOT.PSWC	943 *		
D_Q.ANDNOT.PSWN	944 #		
D_Q.ANDNOT.FSWZ	945 *		
D_Q.LEFT	946 #		
D_Q.OR.KEJ	947 *		
D_Q.OR.PSWC	948 *		
p_Q.OR.RCCJ	949 #		
D_Q.ORNOT.MASK	950 #		
D_Q.RIGHT	951 #		
D_Q.RIGHT2	952 #		
D_Q.SXTCI	953 #		
D_Q.XOR.RCEJ	954 #		
p_gc pp	955 #		
D_GCJKCJ	956 #		
DQE JMASK	957 #		
D_R(PRN+1)	958 #		
n_R(SC)	959 #		
D_R(SP1+1)	960 *		
D_RC(SC)	961 #		
D_RCE3	962 #		
D_RLOG	963 #		
D_RLOG.RIGHT	964 #		
D_REJ	965 \$ 1767		
D_REJ(FRAC)	966 *		
D_REJ.AND.KEJ	967 *		
D_REI.OR.KEI	968 *		
D_REJ.ORNOT.KEJ	969 *		
E-FORK	1562 † 1664 †		
EALU.N?	1001 #		

; OLDSAM.MCR ;	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names	Pase	50
EALU.Z?	1665 #		
EALU?	1666 \$		
EALU_D(EXP)	971 *		
EALU_FE	972 *		
EALULKED	973 #		
EALU_RE3(EXP)	974 #		
EALU_SC	975 #		
EALULSC+FE	976 #		
EALU_SC+KEI	977 #		
EALU_SC-FE	978 #		
EALU_SC-KCJ	979 #		
EALULSC.ANDNOT.KEJ	980 #		
EALULSTATE	981 🛊		
END.DP1?	1667 #		
EXCEPT.ACK	1563 #		
FESSCLKCI	983 #		
FE_0(A)	984 #		
FE_D(EXF)	985 #		
FE_EALU FE_KC)	986 * 987 *		
FE_LA(EXP)	988 #		
FE_NABS(SC-FE)	789 - 989 -		
FE_NABS(SC-LA(EXP))	990 #		
FE_Q(EXP)	991 #		
FELREJ(EXP)	992 #		
FE_SC	993 #		
FE_SC+1	994 #		
FE_SC+FE	995 #		
FELSC+KE3	996 #		
FE_SC+LA(EXF)	997 #		
FE_SC-FE	998 #		
FE_SC-KEJ	999 #		
FE_SC-LA(EXF)	1000 #		
FE_SC-SHF.VAL FE_SC.ANDNOT.FE	1001 + 1002 +		
FE_SC.ANDNOT.KEJ	1003 #		
FE_SC.OR.KCJ	1004 #		
FE_SHF.VAL	1005 #		
FE_STATE	1006 #		
FLUSH. IB	1565 #		
FPD?	1669 #		
G.FORK	1567 #		
IB.TEST?	1671 #		
TD(SC)_D	1008 #		
IDC J_D	1009 #		
ID_D&NO.SYNC	1010 #		
ID_D.SYNC	1011 *		
INHIBIT.IB	1569 \$		
INT?	1672 # 1784		
INTERRUFT.REQ?	1673 #		
INTRPT.ACK INTRPT.STROBE	1570 ‡ 1571 ‡ 1776 1837		
IRO.C31?	1674 #		
IRO?	1675 #		
IR1?	1676 ‡		

DLDSAM.MCR	MICRO2 1L(02) Cross Reference	18-JAN-82 16:19:40 Listing - Macro Names
TR2-1?	1677 #	
IRD	1572 #	
IRD.11	1573 #	
TRDO	1574 #	
IRD1	1575 #	
KEJ	1013 #	
LAB_R(DST)	1015 #	
LAB_R(PRN)	1016 #	
LAB_R(PRN+1)	1017 #	
LAB_R(SC)	1018 #	
LAB_R(SF1)	1019 #	
LAB_R(SP1+1)	1020 #	
LAB_R1&RCCJ_O	1021 #	
LAB_R18RCCJ_O+LC+1	1022 #	
LAB_R18RCCJ_O-D	1023 #	
LAB_R1&RCEJ_ALU	1024 # 1771	
LAB_R1&RCEJ_ALU.RIGHT2	1025 #	
LAB_R18RCCJ_D+LC	1026 #	
LAB_R1%RCCJ_D.OXTCJ+KCJ	1027 #	
LAB_R1&RCEJ_Q-KEJ	1028 #	
LAB_REJ	1029 #	
LAST.REF?	1679 #	
LA_R(DST)&LB_R(SRC)	1031 #	
LA_R(SP2)%LB_R(SP1)	1032 #	
LALRACI	1033 # 1808	
LC_RC(SC)	1034 #	
L.CRCE I	1035 # 1783	
LC_RCEJ&R1_(LA+LB).LEFT	1036 #	
LC_RCCJ&R1_(LA+LB+PSL.C).LEF		
LC_RCE3&R1_(LA+LB.RLOG).LEFT		
LC_RCC3&R1_(LA-LB).LEFT	1039 #	
LC_RCC1&R1_(LA-LB.RLOG).LEFT	1040 #	
LC_RCC3&R1_ALU	1041 #	
LC_RCCJ&R1_D	1042 #	
LC_RCCJ&R1_LA+KCJ	1043 #	
LC_RCEJ&R1_LA-KEJ	1045 #	
LC_RCE3&R1_LB LC_RCE3&R1_Q	1046 #	
	1577 #	
LOAD.ACC.CC	1578 #	
LOAD IB	1579 #	•
LOAD.IB.11	1580 # 1788	1807
IONG MEMORY • NOF	1582 #	1007
MODE.LSS.ASTLVL?	1681 #	
MUL.OXT	1583 #	
MUL.1XT	1584 #	
MUL?	1682 #	
MULM . DONE	1585 #	
MULP . DONE	1586 #	
N&Z_ALU	1048 #	
N&Z_ALU.V&C_0	1049 #	
NEST-ERR?	1684 #	
N_AMX+Z_TST	1050 #	
PC&VAALU	1052 #	
PC&VA_D	1053 #	

; OLDSAM.MCR ;	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listing - Macro Names	Pase	52
PC&VA_D+KE3	1054 #		
PC8VA_D-KC3	1055 #		
PC&VA_D-PC	1056 ‡ 1057 ‡		
PC&VA_D.OXT[]	1057 #		
PC&VA_D.OXTE3+PC	1058 *		
PC&VAD.SXTE3+PC PC&VAKE3	1007 #		
PC&VA_PC	1060 #		
PC&VA_Q	1062 #		
PC&VA_Q+PC	1063 #		
PC&VA_Q-D	1064 #		
PC&VA_Q-KED	1065 #		
PC&VA_Q.SXTCJ+PC	1066 #		
PC8VA_RCCI	1067 #		
PC&VA_REJ.ANDNOT.KEJ	1068 #		
PC.MODES?	1686 #		
PC_PC+1	1070 # 1789 1832		
PC_PC+2	1071 #		
PC_PC+4	1072 #		
PC_PC+N	1073 #		
PC_Q+PC	1074 #		
PC_VA	1075 #		
PC_VIBA	1076 \$		
POLY DONE	1598 #		
PSL.C?	1687 \$		
PSL.CC?	1688 #		
PSL.MODE?	1689 #		
PSL·N?	1690 #		
PSL.V?	1691 #		
PSL.Z?	1692 #		
PSL <c>AMX0</c>	1077 #		
PTE.VALID?	1693 #		
Q&VA_ALU	1079 #		
Q&VA_D	1080 #		
Q&VA_D+LC	1081 #		
Q&VA_LA	1082 #		
Q&VA_Q+LB.PC	1083 #		
0317	1695 #		
QD_(Q+LB)D.RIGHT2	1085 #		
GD_(Q+LC)D.RIGHT2	1086 #		
QD_(Q-LB)D.RIGHT2	1087 #		
QD_(Q-LC)D.RIGHT2	1088 #		
QD_QD.RIGHT2 QUAD?	1089 * 1696 *		
Q_(LA+Q).RIGHT	1091 # 1775		
Q_(Q+LB).RIGHT	1092 \$ 1836		
G_0	1093 #		
Q_0+LC+1	1073 #		
Q_O+MASK+1	1095 #		
Q_0+PC.RL0G	1096 #		
Q_0-D	1097 #		
Q_0-K[]	1098 #		
Q_0-LC	1099 #		
G_0-Q	1100 #		
G_ACCEL &SYNC	1101 #		
THE RESERVE OF THE SALE AND SALE OF THE SALE			

OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names	
Q_ALU	1102 #	
Q_ALU(FRAC)	1103 #	
Q_ALU.LEFT	1104 #	
Q_ALU.LEFT2	1105 #	
Q_ALU.LEFT3	1106 #	
G_ALU.RIGHT	1107 #	
Q_ALU.RIGHT2	1108 #	
Q_D	1109 #	
Q_D(FRAC)(B)	1110 #	
Q_D+KC3	1111 #	
Q_D+KE3+1	1112 #	
Q_D+KCJ.LEFT	1113 #	
Q_D+LC	1114 #	
Q_D-KCJ	1115 #	
QD-L.C	1116 #	
Q.,D-Q	1117 #	
G_D.OXTE3	1118 #	
Q_D.OXTED+KED.LEFT	1119 #	
Q_D.OXTEJ.OR.PACK.FF	1120 #	
Q_D.AND.KEJ	1121 *	
Q_D.AND.KCJ.RIGHT	1122 #	
Q_D.AND.KCJ.RIGHT2	1123 #	
G_D.AND.RCE3	1124 #	
Q_D.ANDNOT.RCC3	1125 #	
Q_D.LEFT3	1126 #	
Q_D.OR.KE3	1127 #	
Q_D.OR.RCE3	1128 #	
Q_D.RIGHT	1129 #	
Q_D.RIGHT2	1130 #	
Q_D.SXTCI	1131 #	
Q_D.XOR.Q	1132 #	
Q_DEC.CON_	1133 #	
Q_IB.BDEST	1134 # 1135 #	
Q_IB.DATA	1136 #	
Q_ID(SC)	1137 #	
G_KE3	1138 #	
Q_KEJ+1	1139 #	
Q_KEJ.CTX	1140 #	
Q_KEJ.RIGHT	1141 #	
Q_KCJ.RIGHT2	1142 #	
Q_LA	1143 #	
Q_LA+KE3	1144 #	
Q_LA+Q	1145 #	
Q_LA-KED	1146 #	
Q_LA.AND.KEJ	1147 #	
Q_LA.ANDNOT.RCCJ	1148 #	
Q_LB	1149 #	
Q_LC	1150 #	
Q_NOT.Q	1151 #	
G_NOT+RED	1152 #	
Q_PACK+FP	1153 #	
QPC	1154 #	
Q_Q(FRAC)	1155 #	
Q_Q(FRAC)(B)	1156 #	

; OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names
0_0+D Q_0+KE3	1157 * 1158 * 1820
Q_Q+K[]+1	1159 #
Q_Q+LC	1160 #
Q_Q+PC	1161 #
Q_Q-D	1162 #
Q_Q-D-1	1163 #
Q_Q-KE3	1164 # 1825 1165 #
Q_Q-K[]-1 Q_Q-LC	1166 #
Q_Q-LC-1	1167 #
Q_Q-MASK-1	1168 #
Q_Q.OXTCJ-KCJ	1169 #
Q_Q.OXTEJ.LEFT	1170 #
Q_Q.OXTEJ.OR.D	1171 #
Q_Q.AND.KEJ	1172 #
Q_Q.AND.KCJ.RIGHT	1174 #
Q_Q.AND.KEJ.RIGHT2 Q_Q.AND.RCEJ	1173 # 1176 #
Q_Q.AND.REJ	1175 #
G_Q.ANDNOT.D	1177 #
Q_Q.ANDNOT.KEJ	1178 # 1781
Q_Q.ANDNOT.RCCJ	1179 #
Q_Q.LEFT	1180 #
Q_Q.LEFT2	1181 #
Q_Q.OR.KC3	1182 #
Q_Q.ORNOT.MASK	1183 #
Q_Q.RIGHT Q_Q.RIGHT2	1184 * 1185 *
Q_Q.SXTEJ	1186 #
Q_Q.XOR.KEJ	1187 #
Q_R(PRN).ANDNOT.Q	1188 #
Q_R(PRN+1)	1189 #
Q_R(PRN+1).AND.Q	1190 #
Q_R(SC)	1191 #
Q_R(SRC!1).AND.KE3	1192 # 1193 #
Q_RC(SC) Q_RCCI	1194 #
Q_RCE3(FRAC)	1195 #
Q_REJ	1196 # 1763
Q_REJ(FRAC)	1197 #
Q_REJ.AND.KEJ	1198 #
Q_REJ.AND.KEJ.RIGHT	1199 #
Q_REJ.ANDNOT.KEJ	1200 #
Q_RCJ.OR.KCJ Q_SC	1201 # 1202 #
Q_SHF	1202 #
R(DST)_ALU	1205 #
R(DST)_D	1206 #
R(DST)_D.SXTCJ.RIGHT	1207 #
R(FRN)_O+D.RLOG	1209 #
R(PRN)_ALU	1210 #
R(PRN)_D	1211 #
R(PRN)_D+KEJ.RLOG	1212 #
R(PRN)_D-KEJ.RLOG	1213 #

OLDSAM.MCR	MICRO2 1L(O2) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names
ŷ	Cross Reference Listins - nacro Rames
R(PRN)_D.OR.Q	1214 #
R(PRN)_DCJQ	1215 #
R(PRN)KEI	1216 #
R(PRN)_LA+KEJ.RLOG	1217 #
R(PRN)_LA+Q	1218 * 1219 *
R(PRN)_LA-KEJ•RLOG R(PRN)_LAEJMASK	1220 #
R(PRN)LC	1221 #
R(PRN)_PACK.FP	1222 #
R(PRN)_Q	1223 #
R(PRN)_Q+KE3.RLOG	1224 #
R(PRN)_Q-KEJ.RLOG	1225 #
R(FRN+1)_ALU	1226 # 1227 #
R(PRN+1)_D R(PRN+1)_D.OR.Q	1228 #
R(FRN+1)_KEJ	1229 #
K(PRN+1)_LA	1230 #
R(PRN+1)_LC	1231 #
R(PRN+1)Q	1232 #
R(SC)_ALU	1234 #
R(SC)_D	1235 * 1236 *
R(SC)_KEJ R(SC)_LA	1237 #
R(SC)_LA+D	1238 #
R(SC)_LA-D	1239 #
R(SC)_LC	1240 #
R(SC)_Q	1241 #
R(SF1)_ALU	1243 #
R(SP1)_D	1244
R(SP1)_KED R(SP1)_PACK.FF	1246 #
R(SP1)_Q	1247 #
R(SP1+1)_LC	1248 #
R(SP1+1)_Q	1249 #
R(SRC!1)_ALU	1251 #
R(SRC!1)_D(B)	1252 #
R(SRC)_ALU R(SRC)_D	1253 # 1254 #
R(SRC)_D(B)	1255 #
R(SRC)_D+KCJ.RLOG	1256 #
R(SRC)_D-KEJ.RLOG	1257 #
R(SRC)_LC	1258 #
R(SRC)_Q	1259 #
R6_D+KCJ.RLOG	1261 #
R6_LA+KE3.RLOG R6_LA-KE3.RLOG	1262 # 1263 #
RC(SC)_O-LC	1265 #
RC(SC)_ALU	1266 #
RC(SC)_ALU.RIGHT	1267 #
RC(SC)_D	1268 #
RC(SC)_Q	1269 #
RCE38VA_D+Q	1271 * 1272 *
RCC3_0 RCC3_0+KC3+1	1273 *
RCEI_OfLC+1	1274 #
1 7 10 10 10 10 10 10 10 10 10 10 10 10 10 10	-

; OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listing - Macro Names
RCEJ_O+MASK+1	1275 #
RCEJ_O+MASK+1.RIGHT2	1276 #
RC[]_O-D	1277 #
RCE JALU	1278 #
RCCJ_ALU.LEFT	1279 #
RCCJ_ALU.LEFT2	1280 #
RCCJ_ALU.LEFT3	1281 #
RCC1_ALU.RIGHT	1282 #
RCCJ_ALU.RIGHT2	1283 #
RCCJ_D	1284 #
RCCJ_D(B)	1285 #
RCC3_D+KC3	1286 #
RCED_D-KCD	1287 #
RCEJ_D.OXTEJ	1288 #
RCCJ_D.AND.KCJ	1289 #
RCEJ_D.AND.MASK	1290 #
RCE3_D.ANDNOT.Q	1291 #
RCED_D.CTX	1292 #
RCCJ_D.LEFT	1293 #
RCC3_D.LEFT3	1294 #
RCC3_D.OR.KC3	1295 #
RCC3_D.OR.Q	1296 #
RCC3_D.ORNOT.KC3	1297 #
RCCJ_D.SXTCJ	1298 #
RCEU_KEU	1299 #
RCC3_KC3+1	1300 #
RCC3_KC3.LEFT2	1301 #
RCCJ_KCJ.LEFT3	1302 #
RCC3_KC3.RIGHT2	1303 #
RCEJLLA	1304 #
RCCJ_LA+LB.CTX	1305 #
RCCJ_LA-KCJ	1306 #
RCCJ_LA.AND.KCJ	1307 #
RCEI_LA.CTX	1308 #
RCEI_LB	1309 #
RCC1_LB.LEFT	1310 ‡ 1311 ‡
RCEILLC	1312 #
RCCJ_NOT.Q	1313 #
RCEJ_PACK+FP	1314 #
RCEJ_PC RCEJ_Q	1315 #
RCEJ_Q+1	1316 #
RCEJ_Q+KEJ	1317 #
RCEJ_Q+LC	1318 #
RCEJ_Q+FC	1319 #
RCCJ_Q+FC+1	1320 #
RCCJ_Q-KCJ	1321 #
RCEJ_Q-LC	1322 #
RCCJ_Q-MASK-1	1323 #
RCCJ_Q.OXTCJ	1324 #
RCEJ_Q.AND.KEJ	1325 #
RCEJ_Q.ANDNOT.KEJ	1326 #
RCCJ_Q.LEFT	1327 *
RCCJ_Q.LEFT3	1328 #
RCC3_Q.RIGHT	1329 #
	· · ·

Page 57

	•
# OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40
9	Cross Reference Listins - Macro Names
RCCJ_Q.RIGHT2	1330 #
RCC3_Q.SXTC3	1331 #
RCEL_RLOG.RIGHT	1332 * 1590 *
RETURNO RETURNI	1570 *
RETURN10	1592 #
RETURN100	1593 #
RETURN10C	1594 #
RETURN10E	1595 #
RETURN12	1596 #
RETURN18	1597 #
RETURN1F	1598 +
RETURN2	1599 * 1600 *
RETURN20 RETURN24	1601 #
RETURN3	1602 #
RETURN4	1603 #
RETURN40	1604 #
RETURN60	1605 #
RETURN61	1606 #
RETURNS	1607 #
RETURN9	1609 #
RETURNE	1609 #
RETURNED	1610 #
RLOG.EMPTY?	1698 # 1699 #
ROR? RED&VALA+KED	1334 #
REJ&VA_LA-KEJ	1335 #
REJAVA_LA-KEJ.RLOG	1336 #
REJ8VAQ-KEJ	1337 #
REJ_O	1338 #
REJ_O+LB+1	1339 #
REJ_0-1	1340 #
RE1_0-D	1341 #
REJ_O-KEJ	1342 * 1343 *
REJ_O-LB REJ_O-Q	1344 #
REJLALU	1345 # 1821 1826
RE3_ALU.LEFT	1346 #
REJ_ALU.LEFT3	1347 #
REJ_ALU.RIGHT	1348 #
REJ_ALU.RIGHT2	1349 #
REJLD	1350 #
REDLD+KED	1351 #
RCJ_D+Q	1352 #
RC3_D+Q+1	1353 # 1354 #
RE3_D-KE3 RE3_D-LC-1	1355 #
RC1_D-Q	1356 #
REJ_D.AND.KEJ	1357 #
REJ_D.OR.LC	1358 #
REJ_D.OR.FACK.FF	1359 #
REJ_D.OR.Q	1360 #
REDLKED	1361 #
REJLLA	1362 #

Page 58

RCJ_LA+D RCJ_LA+D+1 RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+KLJ RCJ_LA+CD RCJ_LA+D RCJ_LA-D RCJ_LA-D RCJ_LA-CD RCJ_LA-KLJ RCJ_LA-KLJ RCJ_LA-KLJ RCJ_LA-KLJ RCJ_LA-KLJ RCJ_LA-KLJ RCJ_LA-CD RCJ_LA-MSK-1 RCJ_LA-MSK-1 RCJ_LA-MSK-1 RCJ_LA-Q RCJ_LA-MSK,CJ RCJ_LA-ND,RCJ RCJ_LA-ND,RCJ RCJ_LA-RCJ,RCJ RCJ_LA-RCJ,RCJ RCJ_LA-RCJ,RCJ RCJ_LCJ RCJ_LCJ RCJ_LCJ RCJ_LCJ RCJ_LCJ RCJ RCJ_LCJ RCJ RCJ_LCJ RCJ RCJ_LCJ RCJ RCJ_LCJ RCJ RCJ RCJ_LCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ R	; OLDSAM.MCR ;	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names
RCJ_LA+D+1 RCJ_LA+KCJ+1 RCJ_LA+KCJ+1 RCJ_LA+KCJ+1 RCJ_LA+KCJ+1 RCJ_LA+CC RCJ_LA+LC RCJ_LA+LC RCJ_LA+LC RCJ_LA+LC RCJ_LA+LC RCJ_LA+LC RCJ_LA+CC RCJ_LA+CC RCJ_LA+CC RCJ_LA+CC RCJ_LA+CC RCJ_LA-CC RCJ_LC-CC RCJ_LC RCJ_LC RCJ_LC RCJ_LC RCJ_LC RCJ_LC RCJ_LC RCJ_CC RCJ_LC RCJ_CC R	REJ_LA+D	1363 #
REJ_LAHKEJH1 1366		
REJ_LAHKEJ+I REJ_LAHLOR REJ_LAHLOR REJ_LAHLOR REJ_LAHLOR REJ_LAHCR REJ_LC REST REJ_LONOT. NASK REJ_REST REJ_REST REJ_LONOT. NASK REST REJ_LONOT. NASK REST REJ_LONOT. REST RESTRICT-RELJ(EXP) REJ_LONOT. REST RESTRICT-RELJ(EXP) RESTRICT-RELJ(EXP) RES_LONOT. RESTRICT-RELJ(EX	REJ_LA+KEJ	1365 #
RCJ_LA+KCJ.RLOG RCJ_LA+HC RCJ_LA+HC RCJ_LA+HC RCJ_LA+C RCJ_LA+C RCJ_LA+C RCJ_LA-C RCJ_C RCJ_LC RCJ_C R		1366 #
REJ_LA+LC REJ_LA+MASK+1 13.69		1367 #
REJLLA-D REJLA-KIJ 1370		1368 #
REJ_LA-D REJ_LA-KI] 1372	REJ_LA+MASK+1	1369 #
RCJ_LA-KCJ RCJ_LA-KCJ,RLOG RCJ_LA-MASK-1 RCJ_LA-MASK-1 RCJ_LA-MAD.KCJ RCJ_LA-OR.D RCJ_LA.OR.D RCJ_LA.OR.D RCJ_LA.OR.D RCJ_LC RCJ_RCC RCC RCJ_RCC RCC RCJ_RCC RCC RCJ_RCC RCC RCC RCC RCC RCC RCC RCC RCC RCC	REJ_LA+Q	1370 #
RCJ_LA-KCJ.RLOG	REJ_LA-D	1371 #
RCJ_LA-MASK-1	REJ_LA-KEJ	1372 #
REJLA-Q REJLA-AND.KEJ REJLA.AND.KEJ REJLA.AND.KEJ REJLA.ORNOT.MASK 1378	REJLLA-KEJ+RLOG	1373 #
REJLA.AND.KEJ 1376 REJLLA.ORNDT.MASK 1378 REJLLE 1379 REJLLC 1380 REJLLC RIGHT 1381 REJLLC.RIGHT 1381 REJLNOT.O 1382 REJLNOT.MASK 1384 REJLNOT.MASK 1384 REJLNOT.MASK 1384 REJLNOT.MASK 1385 REJLNOT.MASK 1385 REJLNOT.MASK 1386 REJLOHT 1380 REJLOHT 1380 REJLOHT 1380 REJLOHT 1380 REJLOHT 1388 REJLOHT 1388 REJLOHT 1388 REJLOHT 1388 REJLOHT 1388 REJLOHT 1389 REJLOHED 1388 REJLOHED 1389 REJLOHED 1400 REJLOHED 1400	REJ_LA-MASK-1	1374 #
RCJ_LA.OR.D	REJ_LA-Q	1375 #
RCJ_LA.ORNOT.MASK	REJ_LA.AND.KEJ	1376 #
RCJ_LB	REJ_LA.OR.D	1377 #
RCJ_LC.RIGHT 1380	REJ_LA.ORNOT.MASK	1378 #
RIJ_LC.RIGHT	REJLLB	1379 #
RCJ_NOT.D RCJ_NOT.D RCJ_NOT.MASK RCJ_NOT.Q RCJ_NOT.Q RCJ_PACK.FP 1384	REJLLC	
RCJ_NOT.D RCJ_NOT.MASK RCJ_NOT.MASK RCJ_NOT.MASK RCJ_PACK.FP RCJ_Q RCJ_G+CK_FP RCJ_Q RCJ_G+S RCJ_G+S RCJ_G+S RCJ_G+S RCJ_G+S RCJ_G+S RCJ_G+S RCJ_G+S RCJ_G+S RCJ_G-S R		
REJ_NOT.MASK REJ_NOT.Q REJ_PACK.FF REJ_Q REJ_Q REJ_G REJ		
RCI_PACK.FP		
RCJ_PACK.FP RCJ_Q RCJ_Q+1 RCJ_Q+1 RCJ_Q+5 RCJ_Q+KCJ RCJ_Q+KCJ RCJ_Q+KCJ RCJ_Q-D RCJ_Q-D RCJ_Q-D-1 RCJ_Q-D-1 RCJ_Q-KCJ RCJ_Q-RCJ RCJ_RCJ RCJ_RCJ RCJ_RCJ RCJ_RCJ RCJ RCJ_RCJ RCJ RCJ_RCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ		
RED_Q RED_Q+1 RED_Q+5 RED_Q+S RED_Q+KED RED_Q+KED RED_Q+LB RED_Q+LC RED_Q+CD RED_Q+KED		
RCI_Q+1		
RCI_Q+S RCI_Q+KCI RCI_Q+KCI RCI_Q+LC RCI_Q-D RCI_Q-D RCI_Q-D RCI_Q-D-1 RCI_Q-D-1 RCI_Q-NCI RCI_Q-KCI RCI_Q-KCI RCI_Q-KCI RCI_Q-KCI RCI_Q-KCI RCI_Q-KCI RCI_Q-KCI RCI_Q-KCI RCI_Q-KCI RCI_Q-CC RCI_Q-CC RCI_Q-CC RCI_Q-CC RCI_Q-CC RCI_Q-RNDNOT.KCI RCI_Q-RNDNOT.KCI RCI_Q-RNDNOT.KCI RCI_Q-RCRNOT.KCI RCI_Q-RCRNOT.KCI RCI_Q-RCROT. RCI_Q-	• • • • • • •	
RCJ_Q+KCJ		
RCI_Q+LB RCI_Q+LC RCI_Q-D RCI_Q-D RCI_Q-D-1 RCI_Q-D-1 RCI_Q-KCI		
RCI_Q-D RCI_Q-D RCI_Q-D RCI_Q-D-1 RCI_Q-KCI RCI_Q-C RCI_Q-		
RCJ_Q-D RCJ_Q-NCJ RCJ_Q-KCJ RCJ_Q-KCJ RCJ_Q-KCJ RCJ_Q-KCJ RCJ_Q-LC RCJ_Q-LC RCJ_Q-LC RCJ_Q-LC RCJ_Q-LC RCJ_Q-LC RCJ_Q-RCJ RCJ_Q-RONOT.KCJ RCJ_Q-RONOT.KCJ RCJ_Q-RORNOT.KCJ RCJ_Q-RCR,D RCJ_Q-RCR,D RCJ_Q-RCR,C RCJ_Q-RCR,C RCJ_Q-RCR,C RCJ_RCG,C RCJ_RCG,C RCJ RCJ_RCG,C RCJ RCJ_RCG,C RCJ RCJ_RCG,C RCJ RCJ RCJ_RCG,C RCJ RCJ RCJ_RCG,C RCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ RCJ RC		
RCJ_Q-NCJ RCJ_Q-KCJ RCJ_Q-KCJ RCJ_Q-KCJ RCJ_Q-LC RCJ_Q-AND.KCJ RCJ_Q.AND.KCJ RCJ_Q.AND.KCJ RCJ_Q.OR.D RCJ_Q.OR.D RCJ_Q.OR.D RCJ_Q.OR.D RCJ_Q.RCGHT.1 1402 \$ RCJ_RCG.RCGHT.1 1403 \$ SC_SCTATE_STATE—RCJ(EXF) 1405 \$ SC_SCT.O? 1701 \$ SC_NE.O? 1702 \$ SC_O-KCJ SC_O-KCJ SC_ALU SC_ALU SC_ALU SC_ALU SC_D(EXF) 1411 \$		
RCI_Q-KCJ.RLOG		
REJ_Q-LC	REJ_Q-KEJ	1395 #
RCILG.AND.KCI 1398	RCJ_Q-KCJ.RLOG	1396 #
RED_Q.ANDNOT.KED 1399	REJ_Q-LC	1397 #
REJ_Q.OR.D REJ_Q.ORNDT.KEJ 1401	RE3_Q.AND.KE3	1398 #
RCI_Q.ORNOT.KCI		
RCJ_Q.RIGHT.1		
RCJ_RLOG.RIGHT.1 1403 # SC2STATE_STATE_RCJ(EXF) 1405 # SC.OF.O? 1701 # SC.OR 1702 # SC.OR 1703 # SC_O(A) 1406 # SC_O-KCJ 1407 # SC_ALU 1408 # SC_ALU 1409 # SC_ALU(EXP) 1409 # SC_D 1410 # SC_D(EXF) 1411 #		
SC&STATE_STATE—REJ(EXF) 1405		
SC.GT.0? 1701 # SC.NE.0? 1702 # SC.? 1703 # SC.O(A) 1406 # SC.O-KEJ 1407 # SCALU 1408 # SCALU(EXP) 1409 # SCD 1410 # SCD(EXP) 1411 #		
SC.NE.0? 1702 # SC? 1703 # SC_O(A) 1406 # SC_O-K[] 1407 # SC_ALU 1408 # SC_ALU(EXP) 1409 # SC_D 1410 # SC_D(EXP) 1411 #		
SC? 1703 # SC_O(A) 1406 # SC_O-K[] 1407 # SC_ALU 1408 # SC_ALU(EXP) 1409 # SC_D 1410 # SC_D(EXP) 1411 #		
SC_O(A) 1406		
SC_O-KE] 1407		
SC_ALU 1408		
SC_ALU(EXP) 1409 # SC_D 1410 # SC_D(EXP) 1411 #		
SC_D 1410 \$ SC_D(EXF) 1411 \$		
SC_D(EXP) 1411 #		
SC_D(EXP)(B) 1413 #		
SC_D-K[] 1414 ‡		
SC_D.OXTEJ-KEJ 1415 #	SC_D.OXTE3-KE3	1415 #

Pase 59

; OLDSAM.MCR	MICRO2 1L(02) Cross Reference	18-JAN-82 16:19:40 Listing - Macro Names
SC_D.OXTEJ.XOR.KEJ	1416 #	
SC_D.AND.KEJ	1417 #	
SC.D.OR.KCJ	1418 #	
SC_D.SXTED	1419 #	
SCLEALU	1420 # 1421 #	
SC_FE	1422 #	
SC_KEJ.ALU	1423 #	
SCLA	1424 #	
SC_LA.AND.KEJ	1425 #	
SCLC(EXP)	1426 #	
SC_NABS(SC-FE)	1427 #	
SC_PSLADDR	1428 #	
SC_Q	1429 #	
SC_Q(EXP)	1430 #	
SC_Q(EXP)(B)	1431 #	
SC_Q+KCI	1432 #	
SC_Q-KED	1433 #	
SC_Q.AND.KEJ	1434 #	
SC_Q.OR.KED	1435 #	
SC_Q.SXTE3	1436 #	
9C_RCE3	1437 #	
SC_RCED(EXP)	1438 #	
SCLRED	1439 # 1440 #	
SC_REJ(EXP)	1441 #	
SC_RED.AND.KED	1442 #	
SC_SC+1 SC_SC+EXP(Q)(A)	1443 #	
SC_SC+FE	1444 #	
SC_SC+KET	1445 #	
SC_SC+SHF.VAL	1446 #	
SC_SC-FE	1447 #	
SC_SC-KCI	1448 #	
SC_SC-SHF.VAL	1449 #	
SC_SC.ANDNOT.FE	1450 #	
SC_SC.ANDNOT.KED	1451 #	
SCLSC.OR.KEJ	1452 #	
SC_SHF.VAL	1453 #	
SC_STATE	1454 #	
SC_STATE.ANDNOT.KCJ	1455 #	
SC_STATE.OR.KEJ	1456 #	
SD_NOT.SD	1457 #	
SDLSS	1458 # 1612 #	
SET.CC(BYTE)	1613 #	•
SET.CC(INST)	1614 #	
SET.CC(LONG) SET.CC(ROR)	1615 #	
SET.CC(WORD)	1616 #	
SET.FPD	1617 #	
SET.NEST.ERR	1618 #	
SET.PSL.C(AMX)	1619 #	
SET.V	1620 #	
SIGNST	1704 #	
SPEC	1621 #	
SPECG	1622 #	

; OLDSAM.MCR	MICRO2 1L(O2) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names	Pase	60
SRC+PCP	1705 ‡		
SS?	1706 #		
SS_0&SD_0	1459 #		
SS_ALU15	1460 #		
SS_SD	1461 #		
SS_SS.XOR.ALU15&SD_ALU15	1462 #		
START. IB	1623 #		
STATE(7)?	1707 #		
STATEO?	1708 # 1777 1838		
STATE1-0?	1709 #		
STATE1?	1710 #		
STATE2?	1711 #		
STATE3-0?	1712 #		
STATE3?	1713 #		
STATE4?	1714 #		
STATES?	1715 #		
STATE6?	1716 #		
STATEZ-4?	1717 #		
STATE_O(A)	1463 #		
STATE_AMX.EXF	1464 *		
STATE_D(EXP)	1465 #		
STATE_FE	1466 #		
STATE_FIRST	1467 *		
STATE_INNEROBJ	1468 *		
STATE_INNERSRC	1469 #		
STATE_KEJ	1470 # 1764 1815		
STATE_OUTER	1471 #		
STATELPREDEC	1472 #		
STATE_Q(EXP)	1473 #		
STATE_SC.VIA.KMX	1474 #		
STATE_SKPLONG	1475 #		
STATE_STATE+1	1476 #		
STATE_STATE+FE	1477 #		
STATE_STATE+KEJ	1478 #		
STATE_STATE-FE	1479 #		
STATE_STATE-KCJ	1480 #		
STATE_STATE.AN.5TOO	1482 #		
STATE_STATE.AN.6T04	1483 #		
STATE_STATE.AN.DESTDBL	1484 #		
STATE_STATE.AN.NOTPREDEC	1485 #		
STATE_STATE.AN.PREDECZERO	1496 #		
STATE_STATE.AN.SKFLONG	1481 #		
STATE_STATE.ANDNOT.FE	1487 #		
STATE_STATE . ANDNOT . KCJ	1488 #		
STATE_STATE.ANDNOT.SHF.VAL	1489 #		
STATE_STATE.OR.ADJINF	1492 #		
STATE_STATE.OR.DEST	1493 #		
STATE_STATE.OR.DESTDBL	1494 #		
STATE_STATE.OR.FE	1490 #		
STATE_STATE.OR.FILL	1495 #		
STATE_STATE.OR.FLOAT	1496 *		
STATE_STATE.OR.KEJ	1470 *		
STATE_STATE.OR.MOVE	1497 #		
STATE_STATE.OR.PATT1	1498 #		
STATE_STATE.OR.PATT2	1499 #		
NOTES TO THE COLUMN TEST OF THE THE THE TEST TO SEE	*****		

Page 61

; OLDSAM.MCR	MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Macro Names
STOP.IB	1624 #
SWAPD	1500 #
TB.TEST?	1719 #
TEST. TB.RCHK	1626 #
TEST. TB. WCHK	1627 #
TRAP.ACCEJ	1628 #
VA31-30?	1721 #
VA31?	1722 #
VA_ALU	1502 # 1782
VA_D	1503 #
VA_D+KCJ	1504 #
VA_D+LC	1505 #
VA_D+Q	1506 #
VAD.OXTE3+Q	1507 #
VAD.ANDNOT.KCJ	1508 #
VA_KED	1509 #
VALA	1510 #
VA_LA+D	1511 #
VA_LAHKEJ	1512 #
VA_LA+KEJ+1	1513 #
VALLATEC	1514 #
VALA+Q	1515 #
VA_LA-D	1516 *
VA_LA-KEJ	1517 #
VALLA-KEJ-1	1518 # 1519 #
VA_LA-Q	1520 #
VALLA.AND.LC	1520 *
VA_LA.ANDNOT.KEJ VA_LB+D.OXT	1522 *
VA_FC	1523 #
VAQ	1524 #
VA_Q+D	1525 #
VA_Q+KE3	1526 #
VA_Q+LB	1527 #
VA_Q+LB.FC	1528 #
VA_Q+LC	1529 #
VA_Q+PC	1530 #
VA_Q-KEJ	1531 #
VAQ-LB	1532 #
VA_Q.ANDNOT.KEI	1533 #
VALRCEI	1534 #
VA_RE3	1535 #
VA_VA+4	1536 #
WORD	1630 #
WRITE.DEST	1631 #
WRITE.G.DEST	1632 #
Z?	1724 # 1809
ZONED?	1725 #

OLDSAM.MCR

MICRO2 1L(02) 18-JAN-82 16:19:40 Cross Reference Listins - Expression Names

Pase 62

; OLDSAM.MCR ;			ICRO2 10 ocation /	_(02) / Line No	-NAU-81 nl rødmu		9:40		Pase	63
#Location	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F		
U 0000 - 13FF U 1400 U 1408	Unused 1784≔ 1771	1790= 1777	1812= 1822=	1816= 1827=	1764 1809	1767 1838	1796=	1799= 1833=		

OLDSAM.MCR U-code Microword Summary 16:19:40

BSERCH 1400-17FF in bounds
VAXDEF 0 0
BSERCH 15 0

0

Pase 64

Total microwords used in memory U: 15 Total microwords remaining in memory U: 1009 Highest address used in memory U: 140F (hex)

15 1009

Used

Remaining

; OLDSAM.MCR MICRO2 1L(02) 18-JAN-82 16:19:40 Page 65

Pass 1 warnings: 0 Pass 2 warnings: 0 Pass 1 errors: 0 Pass 2 errors: 0

C.3 THE OBJECT FILE (.ULD)

```
FRTOL
FRADIX 16
[1404]=0000003C19C0FA1014047405
E14053=0800003C0180FA0000001408
E14083=0001003C0180FB0800001409
C14093=005C171401C0F80040001400
[1400]=00192E240DC0F90802001406
£14013=C01800381980F80440500062
E14063=001C0020018042100010140C
[1407]=0000003C0180F800000004F8
E140C3=001101000180F88800101402
E14023=0001B3C01B0F88800101402
E14023=00001B3C01B0F8000000140A
E14033=00001B3C0580F8001404740A
E140A3=0019201411C0FA880000140D
E140B3=0019200011C0FA9000001409
C140F3=C001203C0180FA8C40500062
E140D]=004D371401C0F80040001400
FIELD ACF=<71:70>
CONTROL=3
 NOP=0
SYNC=1
 TRAP=2
FIELD ACM=<57:55>
ABORT=1
 POLY.DONE=6
FWR.UP=0
FIELD ADS=<47:47>
 IBA=1
 UA≔0
FIELD ALU=<69:66>
 A≕OF
 A+B=5
 A+B+1=4
 A+B+PSL.C=OB
 A+B.RLOG∞6
 A-B=0
 A-B-1=2
 A-B-RLOG=1
AND=OD
 ANDNOT=9
 B≕OE
 INST.DEP=3
 NOTA=0A
 OR≔OC
 ORNOT=7
 XOR≔8
FIELD AMX=<81:80>
 LA=0
 RAMX#1
 RAMX.OXT=3
 RAMX.SXT=2
FIELD BEN=<76:72>
 ACCEL=6
 ALU=1B
```

ALU1-0=15

```
C31=3
D.BYTES=18
D3-O=19
DATA.TYPE=8
DECIMAL=OF
  EALU=12
  END.DP1=8
  IB.0=5
IB.TEST=OB
  INTERRUPT=0E
 IR2-1=9
IRC.ROM=4
 LAST.REF=11
MUL=0C
  NOF'=0
  PC.MODES=9
 PSL.CC=1A
PSL.MODE=1C
  REI=OA
 ROR≔2
 SC=14
SIGNS=OD
SRC.PC=OA
STATE3-O=17
STATE7-4=16
TB.TEST=1F
 Z=1
FIELD BMX=<84:82>
 KMX≕6
  LB≔3
  L.C = 4
 MASK=0
PACKED.FL=2
 PC=5
PC.OR.LB=1
RBMX=7
FIELD CCK=<22:20>
C_AMX0=6
 INST.DEP=Z
LOAD.UBCC=1
 NOP=0
NZ_ALU.VC_0=5
 NZ_ALU.VC_VC=6
N_AMX.Z_TST.VC_VC=3
ROR=4
SET.V=2
FIELD CID=<45:42>
ACK=5
  CONT=7
  NOF≔1
 READ.KMX=OB
  READ.SC=9
  WRITE.KMX=OF
 WRITE.SC=OD
FIELD DK=<91:88>
ACCEL=0A
  BYTE.SWAP=0B
 CLR=OF
DAL.SC=OD
DAL.SV=OE
 DIV=4
LEFT=5
  LEFT2=1
```

```
NOP=0
Q=0C
RIGHT=6
RIGHT2=2
SHF=8
SHF.FL=9
FIELD DT=<79:78>
BYTE=2
INST.DEP=3
  LONG=0
WORD=1
FIELD EALU=<15:13>
  A+1=6
  A+B=4
  A-B=5
ANDNOT=2
  B = 3
 NABS.A-B=7
OR=1
FIELD EBMX=<19:18>
  AMX.EXF=2
FE=0
KMX=1
SHF.VAL=3
FIELD FEK=<24:24>
LOAD=1
NOP=0
FIELD FS=<42:42>
 CID=1
 MCT=0
MCT=0
FIELD IBC=<95:92>
BDEST=7
CLR.0=0C
CLR.0=3=0E
CLR.0-1=4
CLR.1=0D
CLR.1-5.COND=0F
CLR.2.3=5
EINGH=2
  FLUSH=2
  NOF=0
  START=3
STOP=1
FIELD ID.ADDR=<63:58>
ACC.0=14
ACC.1=15
ACC.2=16
  ACC.CS=17
  CES=OC
  CLK.CS=0A
  COMP=10
  D.SV=2E
  DAY.TIME=1
 ESP=29
FAULT=1B
FPDA=2D
  IBUF=0
  INTERVAL=0B
  ISP=2C
  KSP=28
  MAINT=1D
NXT.PER=9
```

POBR=24

```
POLR=3C
 P1BR≈25
 PILR=3D
PARITY=1E
PCBB=3A
 PSL=0F
 Q.SV=2F
 RXCS=4
 RXDB≈5
 SBI.ERR=19
SBR=26
SCBB=3B
SILO=18
SIR=0E
 SLR=3E
 SSP=2A
 SYS.ID=3
 TO=30
 T1=31
 T2=32
 T3=33
T4=34
T5=35
 T6=36
T7=37
 T8=38
 T9=39
 TRERO=12
 TBER1=13
 TBUF=10
 TIME.ADDR=1A
 TXCS=6
TXDB=7
 UBREAK=21
USP=2B
 USTACK=20
 VECTOR#OD
 WCS.ADDR=22
 WCS.DATA=23
FIELD IEK=<31:30>
EACK=3
 IACK=2
ISTR≕1
NOP≔0
ADDRESS J=<12:0>
INT.B=4F8
 IRD=62
 SRCH=1404
 SRCH.1=1409
SRCH.2=1400
 SRCH.3=1406
 SRCH.4=140C
SRCH.5=140A
SRCH.6=140D
FIELD KMX=<63:58>
.1=1
.10=19
.14=8
.18=1F
 •19=2E
 .1A=39
 .1B=3B
.1E=14
```

```
.1F=23
.1F00=24
  .2=2
.20=1D
  .24=3A
.28=0B
  .3=3
.30=1E
  .3030=32
  .3030=3.
.34=0A
.3F=15
.3FF=20
.4=4
.40=0C
 .4000=20
.50=01
.6=35
  .60=29
.7=17
.70=27
  .7E=3E
  .7F=16
.7FF0=0E
.8=0
  .80=10
  .8000=11
  .88=31
  .9=36
  •A≔3D
  .A0=9
  .B0=25
  .C=21
  .CO=34
  .D=22
.DFCF=2R
.E003=26
.EF=0F
.F=18
  .F0=33
.FF=12
.FF00=13
  .FFE0=28
  .FFE8=1A
.FFF0=1B
.FFF1=2D
.FFF5=38
  .FFF6=37
  .FFF8=1C
  •FFF9=2F
  .FFFC=3C
  .FFFF=30
 SC=7
SF1.CON=5
  SP2.CON≕6
ZERO=6
FIELD MCT=<47:42>
ALLOW.IB.READ=3E
EXTWRITE.P=28
  INVALIDATE=24
 LOCKREAD.P=3A
  LOCKREAD.V.NOCHK=1A
  LOCKREAD.V.WCHK=1C
 LOCKWRITE.P=2E
```

```
LOCKWRITE.V.XCHK=0E
  MEM.NOP=2
  READ.INT.SUM=36
 READ.P=32
READ.V.IBCHK=16
READ.V.NEWFC=18
READ.V.NOCHK=12
READ.V.RCHK=10
  READ.V.WCHK=14
 SBI.HOLD=20
SBI.HOLD+UNJAM=22
  TEST.RCHK=0
  TEST.WCHK=4
 VALIDATE=26
WRITE.P=2A
WRITE.V.NOCHK=0A
WRITE.V.WCHK=0C
FIELD MSC=<29:26>
CHK.CHM=1
CHK.FLT.OFR=2
 CHK.ODD.ADDR=3
 CLR.FPD=8
CLR.NEST.ERR=0A
  INH.CM.ADDR=OF
 IRD=4
LOAD.ACC.CC=6
 LOAD.STATE=5
NOP=0
 READ.RLOG=7
RETRY.NO.TRAP=OD
RETRY.TRAF=OE
SECOND.REF=OC
 SET.FPD=9
 SET.NEST.ERR=OB
FIELD PCK=<34:32>
NOP=0
 PC+1=4
PC+2=5
 PC+4=6
PC+N=7
 PC_IBA=2
 PC..VA≔1
VA+4=3
FIELD QK=<54:51>
 ACCEL=OR
CLR=OF
D=OC
 DEC.CON=OA
  ID=OE
 LEFT=5
 LEFT2#1
 NOF'=0
 RIGHT=6
 RIGHT2=2
 SHF≅8
 SHF.FL=9
FIELD RAMX=<77:77>
 D≕O
 Q=1
FIELD RBMX=<77:77>
 D=1
 O≔O
```

FIELD SCK=<23:23>

```
LOAD=1
NOP=0
FIELD SGN=<50:48>
  ADD.SUB=6
CLR.SD+SS=7
 LOAD.SS=1
NOF=0
NOT.SD=3
  SD.FROM.SS=4
SD.FROM.SD=2
SS.FROM.SD=2
SS.XOR.ALU=5
FIELD SHF=<87:85>
ALU=0
ALU.DT=3
LEFT=1
  LEFT3≈5
  RIGHT=2
  RIGHT2=4
FIELD SI=<57:55>
  ASHL.≈2
  ASHR≔1
 DIV=5
DIVD=0
MUL+=6
MUL-=7
ZERO=3
FIELD SMX=<17:16>
 ALU=2
ALU.EXP=3
EALU=0
FE=1
FIELD SPO=<41:35>
 LOAD.LC.SC=6
 NOP=0
WRITE.RC.SC=7
FIELD SP0.AC=<41:38>
 LOAD.LA=2
LOAD.LAB=1
WRITE.RAB=3
FIELD SPO.ACN=<37:35>
PRN=3
  PRN+1=4
  SC=5
SP1+1=6
  SP1.SP1=0
 SP2.SP1=2
SP2.SP2=1
FIELD SP0.ACN11=<37:35>
  DST.DST=1
  DST.SRC=2
  SC=5
 SRC.OR.1=4
SRC.SRC=0
FIELD SP0.R=<41:39>
 LOAD.LAB=4
LOAD.LAB1.WRITE.RC=6
 LOAD.LC=2
LOAD.LC.WRITE.RAB1=7
WRITE.RAB=5
WRITE.RC=3
FIELD SPO.RAB=<38:35>
AP=0C
  FP=0D
```

```
R0=0
R1=1
R15=0F
R2=2
R3=3
R4=4
R5=5
R6=6
R7=7
SP=0E
FIELD SPO.RC=<38:35>
LC.SV=8
MBIT.VA=0F
PC.SV=0C
FTE.MASK=0F
PTE.VA=0A
SC.SV=0D
T0=0
T1=1
T2=2
T3=3
T4=4
T5:5
16=6
T7=7
VA.REF=0E
VA.SU=9
FIELD SUB=<65:64>
CALL=1
NOP=0
RET=2
SPEC=3
FIELD VAK=<25:25>
LOAD=1
NOP=0
END
```

APPENDIX D

THE TEST PROGRAM

```
•TITLE BSTEST - PROGRAM TO EXERCISE BSERCH TEST MICROCODE •PSECT BSTEST
#Open the terminal for output
        PUSHL
BEGIN:
                  #1
                                               #Allow writes
         PUSHL.
                   #0
                                               #No name
                                               #Name length = 0
         PUSHL
                   #0
                                               FTTY chanel
         PHSHI
                   #-1
                   #4,FILOPN
                                               FOren terminal for output. 4 rarameters
         CALLS
#Save current XFC SCB vector. Set XFC vector to 2 for access to user microcode
#CMKRNL_S ST_VEC #Change mode to kernal % set vector.
         BLBS
                 RO, INITA
                                               #Branch if no error setting vector.
         $EXIT_S RO
                                               #Exit with error status.
finitialize each longword in ARRAY with with its address.
INITA: CLRL
                  RO
         MOVAL.
                   ARRAY, R1
24:
         MOVL
                   R1,(R1)+
         ADBLSS #1000,R0,2$
$Start with RO equal to the address of ARRAY minus one. Do binary search on
PARRAY and check that search produced the correct result. Increment RO by one pand re-search ARRAY, checking the results, until RO is one more than the
Phishest value in ARRAY
                   ARRAY-1,RO
                                               FINIT COMPARAND
         HAUGH
L.00P #
         MOVAL
                   ARRAY,R1
                                               FLOWER BOUND
                                               JUPPER BOUND
         MOVAL.
                   ARRAY+3996,R2
                                               FINVOKE THE SPECIAL MICROCODE FBRANCH IF NO MATCH FOUND
         .BYTE
                   ^XFC
         BEQL
                   NOMCH
    Match. See if it should have matched.
MATCH: BITL
                   #3,R0
                                               #Should have matched if RO is
         BE.QL.
                   R1CHK
                                               Flonsword allisned.
         PUSHAB
                                               #Push address of error message #Push length of error message
                  BULL
                   BDFNDL
         PUSHL
                   ENDIT
                                               )and so report error
         ERW
    Match. See if R1 has the correct value.
                                               ISEE IF R1 HAS THE CORRECT VALUE
R1CHK: CMPL
                  RO,(R1)
         BEQL
                   BUMP
                                               FALL OK
                                               Push address of error message
         PUSHAB
                  BDADD
         PUSHL
                  BDADDL
                                               Frush length of error message
         BRW
                  ENDIT
                                               jand so report error
    No match. See if it should not have matched.
                  #3,R0
BUMP
NOMCH:
                                               #Should not match if RO is not
         BITL
         BNEO
                                               Flongword alligned.
                                               #Push address of error message
         PUSHAR
                  NOMAT
         PUSHL.
                   NOMATL
                                               Push length of error message
         BRW
                   ENDIT
                                               fand so report error
fincrement RO and branch to top if not done.
BUMP: ADBLEQ #ARRAY+3997,RO,LOOP
```

```
. PAGE
fAll done with no errors. Report successful completion.
        PUSHAB DONE
                                           #Fush address of completion message
        PUSHL
                DONEL.
                                           #Push length of completion message
Doubeut ending message, restore original XFC vector, and exit.
ENDIT: PUSHL CALLS
                #-1
#3,FILOUT
        PUSHL
                 ‡---1
        CALLS #1,FILCLS
$CMKRNL_S RSTOR
                                           #Change mode to kernal & restore vector
        SEXIT_S RO
                                           ≠Exit with status
#Routine to save and set a new XFC SCB vector.
ST_VEC: .WORD
        MOVL
                 EXE#GL_SCB,R4
                                           #R4 sets SCBB
        MOVL.
                 "X14(R4),OLDV
                                           #Save the vector at SCBB+14(hex).
        MOVL.
                 #2,°X14(R4)
                                           #Make the new vector at SCBB+14(hex)=2
        MOVL
                 #1,R0
                                           #Indicate success and
        RET
                                           freturn
#Routine to restore original XFC SCB vextor.
RSTOR: .WORD
                 0
                 EXE#GL_SCB,R4
        MOVL
                                           #R4 sets SCBB
        HOVL.
                 OLDV, "X14(R4)
#1,R0
                                           #Restore original vector
        MOVL.
                                           findicate success and
        RET
                                           #return
        . PAGE
BDFND: .ASCII
BDFNDL: .LONG
                 "Search reports a match when it should not."
                 .-BDFND
BDADD: .ASCII
                 "Search reports wrong address an match."
BDADDL: .LONG
                .-BDADD
NOMAT: .ASCII
                "Search does not find match when it should."
NOMATL: .LONG
                 *--NOMAT
DONE:
                *BSTEST successful completion.*
        ASCIT
DONEL:
        LONG
                 .-DONE
.PSECT
        ARRAY, LONG
OLDV:
        · LONG
                 0
         . LONG
                 0
        • BLKL
                 1
ARRAY:
        • BLKL
                 1000
                                          FBLOCK OF 1000 LONGWORDS

    LONG

                 0
.END
        BEGIN
```

INDEX

*	Address space, 2-11
in constraints, 2-13	Address-spaces
	disjoint ranges, 2-11
.(Period), 2-9	Allocation
.ADDRESS, 2-5	method of, 2-11
.BIN, 2-15	random, 2-12
.CASE, 2-7	sequential, 2-11
.CCODE, 2-3	Arithmetic functions, 2-7
.CHANGE, 2-15	Assembler
.CREF, 2-15	description of, 2-1
.DCODE, 2-3	functions, 2-1
DEFAULT, 2-5	Assembler user interface, 2-16
.ECODE, 2-3	Assembly
ENDIF, 2-14	microprogram, 2-1
hexadecimal, 2-2	Asterisk characters
.ICODE, 2-3	in constraints, 2-13
.IF, 2-14	2 20
.IFNOT, 2-14	Base
.LIST, 2-15	of numbers, 2-2
LTOR, 2-2	Bit direction, 2-2
.MCODE, 2-3	Bit numbering, 2-2
.NBIN, 2-15	Blocks
NCREF, 2-15	in conditional assembly, 2-15
.NEXTADDRESS, 2-5	Boolean functions, 2-7
.NLIST, 2-15	
.OCODE, 2-3	Case function, 2-7
.OCTAL, 2-2	Changing expression names, 2-15
.PAGE, 2-3	Characters
.PARITY, 2-7	in names, 2-4
.RANDOM, 2-12	Command line
REGION, 2-11	MICLD, 3-5
.RTOL, 2-2	MICRO2, 2-16
SELECT, 2-7	Comments, 2-4
.SEQUENTIAL, 2-11	Communication
.SET, 2-7, 2-15	for programs, 2-14
SHIFT, 2-7	Comparison functions, 2-7
.TITLE, 2-3	Conditional assembly, 2-14
.TOC, 2-3	blocks, 2-15
.UCODE, 2-3	Constraints, 2-12
.VALIDITY, 2-6	characters in, 2-13
/INITIAL, 3-6	inner, 2-13
/LIST, 2-17	terminating, 2-13
/NOLIST, 2-17	Contents
/NOULD, 2-17	adding entry to, 2-3
/ULD, 2-17	field indicator, 2-9
	Continuation character, 2-10
Address sets, 2-12	Controlling listing format, 2-15
	<u>-</u>

Counters, 2-9	qualifiers
in value-definition, 2-6	MICLD, 3-6
,	MICRO2, 2-17
Dafaulta 2 E	
Defaults, 2-5	Functions, 2-7
for jump field, 2-5	MICLD, 3-1
Defining	MICRO2, 2-1
address space, 2-11	
field-names, 2-4	Initialization
value-names, 2-6	pattern, 3-2
Definitions	default, 3-2
field, A-l	Initializing
macro, A-12	WCS, 3-2
Direction	Initialization
of bit numbering, 2-2	pattern
Disjoint ranges	file qualifier, 3-6
in address space, 2-11	Instruction
	XFC, 4-1
Entry vector, 4-2	Interface
Error messages	MICLD, 3-5
MICLD, 3-10	MICRO2, 2-16
Examples	
	Tump field 2 E
using MICLD, 3-6	Jump field, 2-5
using MICRO2, 2-18	
Exceptions, 4-2	Keywords
Execution	.(Period), 2-15
microprogram, 4-1	.BIN, 2-15
Expression-names, 2-7	.CCODE, 2-3
changing, 2-15	.CHANGE, 2-15
setting, 2-15	.CREF, 2-15
Expressions, 2-7	.DCODE, 2-3
expression-names, 2-7	.ECODE, 2-3
field contents indicator, 2-9	.ENDIF, 2-14
function calls, 2-7	.HEXADECIMAL, 2-2
numbers, 2-7	.ICODE, 2-3
predefined names, 2-9	.IF, 2-14
value names, 2-8	.IFNOT, 2-14
•	
Extended function call, 4-1	.LTOR, 2-2
	.MCODE, 2-3
Faults, 4-2	.NBIN, 2-15
Field contents indicator, 2-9	.NCREF, 2-15
Field definitions, A-1	.NLIST, 2-15
Field name, 2-4	.OCODE, 2-3
Field-definitions	.OCTAL, 2-2
form of, 2-4	.PAGE, 2-3
File	.RANDOM, 2-12
multiple input	.REGION, 2-11
MICLD, 3-5	.RTOL, 2-2
MICRØ2, 2-17	.SEQUENTIAL, 2-11
parameters	.SET, 2-15
MICLD, 3-5	.TITLE, 2-3
MICRO2, 2-17	.TOC, 2-3

.UCODE, 2-3	Operands
	function, 2-7
Left-bit, 2-4	Over-loading, 3-3
Listing controls, 2-15	
Listing file	Paging, 2-3
qualifier, 2-17	Parameters
Loader	file
functions, 3-1	MICLD, 3-5
user interface, 3-5	MICRO2, 2-17
Loading	Parity, 2-6
microprogram, 3-3	Parity function, 2-7
	Patching
Macro body, 2-9	entry vector, 4-2
Macro definitions, A-12	Pattern
parameters, 2-10	initialization, 3-2
Macro names, 2-9	Pointer field, 2-5
Memories	Predefined names, 2-9
communication among, 2-13	Predefined language, 1-2
Memory-indicator, 2-3	Predefinitions
Messages	fields
error	VAX 11/780, A-1
MICLD, 3-10	macros
MICLD	VAX 11/780, A-1
error messages, 3-10	Program radix, 2-2
functions, 3-10	Program title, 2-3
user interface, 3-5	
MICRO2	Qualifiers, 2-5
description of, 2-1	.ADDRESS, 2-5
functions, 2-1	.DEFAULT, 2-5
MICRO2 user interface, 2-16	.NEXTADDRESS, 2-5
Microinstructions	.VALIDITY 2-5
continuation character, 2-10	
form of, 2-10	Radix, 2-2
Microprogram	Random allocation, 2-12
assembling the, 2-1	Right-bit, 2-4
executing the, 4-1	0.1
loading the, 3-3	Select function, 2-7
Microwords, 2-11	Separators
creation of, 2-11	MICLD command line, 3-5
field-definitions, 2-4	MICRO2 command line, 2-17
Name of the A	Sequential, 2-11
Names, 2-4	Setting expression-names, 2-15
characters in, 2-4	Shift function, 2-7
macro, 2-9	Sub-programs, 2-2
predefined, 2-9 value, 2-6	Subtitle
	or program, 2-3
Number a field, 2-4	mahla of contouts
Number base, 2-2	Table of contents

Title of program, 2-3 ULD file qualifier, 2-17 User interface MICLD, 3-5 MICRO2, 2-16 Validity, 2-6 Value names, 2-6 use in expressions, 2-8 Value-definitions, 2-6 Vectors entry patching, 4-2 interpretation of, 4-2 Verifying installation of board, 3-1 loading procedure, 3-4 WCS, 1-1 initialization of, 3-2 WCS verification of board, 3-1 Word initialization, 3-2 Word width, 2-4 Writable control store, 1-1

VAX-11/780 Microprogramming Tools User's Guide AA-H306B-TE

Country

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

		The state of the s				
Did yo		in this manual		o, specify	y the e	rror
	umber.					

				-, - 		

Please	indicate the	type of reader	that y	ou most n	early r	cepres
Please	-	type of reader		ou most n	early r	repres
Please	Assembly la	nguage program	mer grammer	•	early r	cepres
Please	Assembly la Higher-leve Occasional	nguage program el language pro programmer (ex	nmer ogrammer operienc	eed)	early r	cepres
Please	Assembly la Higher-leve Occasional User with l	nguage programel language proprammer (ex	nmer ogrammer operienc	eed)	early r	cepres
Please	Assembly la Higher-leve Occasional User with l	nguage programel language proprammer (exittle programmer ogrammer	nmer ogrammer operienc ning exp	eed)	early r	cepres
Please	Assembly la Higher-leve Occasional User with l	nguage programel language proprammer (ex	nmer ogrammer operienc ning exp	eed)	early r	cepres
	Assembly la Higher-leve Occasional User with l Student pro Other (plea	inguage programed language proprogrammer (expression)	nmer ogrammer operience ning exp	eed) Perience		
Name_	Assembly la Higher-leve Occasional User with l Student pro Other (plea	inguage programed language proprogrammer (expression)	nmer ogrammer operience ning exp	eed) perience		
Name_ Organi	Assembly la Higher-leve Occasional User with l Student pro Other (plea	inguage programed language programmer (excittle programmer ase specify)	nmer ogrammer operience ning exp	eed) perienceDate		
Name_ Organi	Assembly la Higher-leve Occasional User with l Student pro Other (plea	inguage programed language proprogrammer (expression)	nmer ogrammer operience ning exp	eed) perienceDate		

	Fold Here	
	Do Not Teur - Fold Here and Staple -	
		FIRST CLASS
		PERMIT NO. 33
BUSINESS REPLY MAIL		MAYNARD, MAS
	RY IF MAILED IN THE UNITED STATES	
Postage will be paid by:		
	digital	
VAX 1925	Current Engineering Andover Street SBURY, Massachusetts 01876	
TEWK	SBURY, MASSACHUSETTS 01876	